KD Chart 2 Reference Manual

[rev.2.0]

Generated by Doxygen 1.3.6

Thu May 10 11:06:24 2007

Contents

1 KD Chart 2 Namespace Index **1**

1.1 KD Chart 2 Namespace List 1

2 KD Chart 2 Hierarchical Index **3**

2.1 KD Chart 2 Class Hierarchy 3

3 KD Chart 2 Class Index **7**

3.1 KD Chart 2 Class List 7

4 KD Chart 2 File Index **9**

4.1 KD Chart 2 File List 9

5 KD Chart 2 Page Index **13**

5.1 KD Chart 2 Related Pages 13

6 KD Chart 2 Namespace Documentation **15**

6.1 BackgroundAttributes Namespace Reference 15

6.2 BarAttributes Namespace Reference 16

6.3 DataValueAttributes Namespace Reference 17

6.4 FrameAttributes Namespace Reference 18

6.5 GridAttributes Namespace Reference 19

6.6 KDChart Namespace Reference 20

6.7 LineAttributes Namespace Reference 25

6.8 MarkerAttributes Namespace Reference 26

6.9 PaintContext Namespace Reference 27

6.10 Palette Namespace Reference 28

6.11 RelativePosition Namespace Reference 29

6.12 TextAttributes Namespace Reference 30

6.13 Ui Namespace Reference 31

7 KD Chart 2 Class Documentation 33

7.1 KDChart::AbstractArea Class Reference 33
7.2 KDChart::AbstractAreaBase Class Reference 45
7.3 KDChart::AbstractAreaWidget Class Reference 51
7.4 KDChart::AbstractAxis Class Reference 61
7.5 KDChart::AbstractCartesianDiagram Class Reference 79
7.6 KDChart::AbstractCoordinatePlane Class Reference 115
7.7 KDChart::AbstractDiagram Class Reference 141
7.8 KDChart::AbstractLayoutItem Class Reference 173
7.9 KDChart::AbstractPieDiagram Class Reference 177
7.10 KDChart::AbstractPolarDiagram Class Reference 214
7.11 KDChart::AbstractProxyModel Class Reference 247
7.12 KDChart::AbstractThreeDAttributes Class Reference 249
7.13 KDChart::AttributesModel Class Reference 252
7.14 KDChart::AutoSpacerLayoutItem Class Reference 268
7.15 KDChart::BackgroundAttributes Class Reference 274
7.16 KDChart::BarAttributes Class Reference 277
7.17 KDChart::BarDiagram Class Reference 280
7.18 KDChart::CartesianAxis Class Reference 326
7.19 KDChart::CartesianCoordinatePlane Class Reference 363
7.20 KDChart::Chart Class Reference 408
7.21 KDChart::DataDimension Class Reference 427
7.22 KDChart::DatasetProxyModel Class Reference 430
7.23 KDChart::DatasetSelectorWidget Class Reference 437
7.24 KDChart::DataValueAttributes Class Reference 440
7.25 KDChart::DiagramObserver Class Reference 448
7.26 KDChart::FrameAttributes Class Reference 452
7.27 KDChart::GlobalMeasureScaling Class Reference 454
7.28 KDChart::GridAttributes Class Reference 457
7.29 KDChart::HeaderFooter Class Reference 460
7.30 KDChart::HorizontalLineLayoutItem Class Reference 479
7.31 KDChartEnums Class Reference 484
7.32 KDTextDocument Class Reference 494
7.33 KDChart::Legend Class Reference 496
7.34 KDChart::LineAttributes Class Reference 533
7.35 KDChart::LineDiagram Class Reference 536

7.36 KDChart::LineLayoutItem Class Reference 587
7.37 KDChart::LineWithMarkerLayoutItem Class Reference 593
7.38 KDChart::MarkerAttributes Class Reference 599
7.39 KDChart::MarkerLayoutItem Class Reference 602
7.40 KDChart::Measure Class Reference 608
7.41 KDChart::PaintContext Class Reference 615
7.42 KDChart::Palette Class Reference 617
7.43 KDChart::PieAttributes Class Reference 620
7.44 KDChart::PieDiagram Class Reference 623
7.45 KDChart::PolarCoordinatePlane Class Reference 665
7.46 KDChart::PolarDiagram Class Reference 698
7.47 KDChart::Position Class Reference 736
7.48 KDChart::PositionPoints Class Reference 745
7.49 PrerenderedElement Class Reference 749
7.50 PrerenderedLabel Class Reference 752
7.51 QAbstractItemView Class Reference 758
7.52 QAbstractProxyModel Class Reference 759
7.53 QFrame Class Reference 760
7.54 QLayoutItem Class Reference 761
7.55 QObject Class Reference 762
7.56 QSortFilterProxyModel Class Reference 763
7.57 QTextDocument Class Reference 764
7.58 QWidget Class Reference 765
7.59 KDChart::RelativePosition Class Reference 766
7.60 KDChart::RingDiagram Class Reference 771
7.61 KDChart::SignalCompressor Class Reference 811
7.62 KDChart::TextArea Class Reference 813
7.63 KDChart::TextAttributes Class Reference 829
7.64 KDChart::TextLayoutItem Class Reference 834
7.65 KDChart::ThreeDBarAttributes Class Reference 844
7.66 KDChart::ThreeDLineAttributes Class Reference 849
7.67 KDChart::ThreeDPieAttributes Class Reference 854
7.68 KDChart::VerticalLineLayoutItem Class Reference 858
7.69 KDChart::Widget Class Reference 863
7.70 KDChart::ZoomParameters Class Reference 879

8 KD Chart 2 File Documentation 881

8.1 KDChartAbstractArea.cpp File Reference 881
8.2 KDChartAbstractArea.h File Reference 885
8.3 KDChartAbstractAreaBase.cpp File Reference 886
8.4 KDChartAbstractAreaBase.h File Reference 887
8.5 KDChartAbstractAreaWidget.cpp File Reference 888
8.6 KDChartAbstractAreaWidget.h File Reference 889
8.7 KDChartAbstractAxis.cpp File Reference 890
8.8 KDChartAbstractAxis.h File Reference 891
8.9 KDChartAbstractCartesianDiagram.cpp File Reference 892
8.10 KDChartAbstractCartesianDiagram.h File Reference 893
8.11 KDChartAbstractCoordinatePlane.cpp File Reference 894
8.12 KDChartAbstractCoordinatePlane.h File Reference 895
8.13 KDChartAbstractDiagram.cpp File Reference 896
8.14 KDChartAbstractDiagram.h File Reference 897
8.15 KDChartAbstractPieDiagram.cpp File Reference 898
8.16 KDChartAbstractPieDiagram.h File Reference 899
8.17 KDChartAbstractPolarDiagram.cpp File Reference 900
8.18 KDChartAbstractPolarDiagram.h File Reference 901
8.19 KDChartAbstractProxyModel.cpp File Reference 902
8.20 KDChartAbstractProxyModel.h File Reference 903
8.21 KDChartAbstractThreeDAttributes.cpp File Reference 904
8.22 KDChartAbstractThreeDAttributes.h File Reference 905
8.23 KDChartAttributesModel.cpp File Reference 906
8.24 KDChartAttributesModel.h File Reference 907
8.25 KDChartBackgroundAttributes.cpp File Reference 908
8.26 KDChartBackgroundAttributes.h File Reference 909
8.27 KDChartBarAttributes.cpp File Reference 910
8.28 KDChartBarAttributes.h File Reference 911
8.29 KDChartBarDiagram.cpp File Reference 912
8.30 KDChartBarDiagram.h File Reference 913
8.31 KDChartCartesianAxis.cpp File Reference 914
8.32 KDChartCartesianAxis.h File Reference 916
8.33 KDChartCartesianCoordinatePlane.cpp File Reference 917
8.34 KDChartCartesianCoordinatePlane.h File Reference 918
8.35 KDChartChart.cpp File Reference 919
8.36 KDChartChart.h File Reference 922

8.37 KDChartDatasetProxyModel.cpp File Reference 923
8.38 KDChartDatasetProxyModel.h File Reference 924
8.39 KDChartDatasetSelector.cpp File Reference 925
8.40 KDChartDatasetSelector.h File Reference 926
8.41 KDChartDataValueAttributes.cpp File Reference 927
8.42 KDChartDataValueAttributes.h File Reference 929
8.43 KDChartDiagramObserver.cpp File Reference 931
8.44 KDChartDiagramObserver.h File Reference 932
8.45 KDChartEnums.h File Reference 933
8.46 KDChartFrameAttributes.cpp File Reference 934
8.47 KDChartFrameAttributes.h File Reference 935
8.48 KDChartGlobal.h File Reference 936
8.49 KDChartGridAttributes.cpp File Reference 941
8.50 KDChartGridAttributes.h File Reference 942
8.51 KDChartHeaderFooter.cpp File Reference 943
8.52 KDChartHeaderFooter.h File Reference 944
8.53 KDChartLayoutItems.cpp File Reference 945
8.54 KDChartLayoutItems.h File Reference 948
8.55 KDChartLegend.cpp File Reference 949
8.56 KDChartLegend.h File Reference 950
8.57 KDChartLineAttributes.cpp File Reference 951
8.58 KDChartLineAttributes.h File Reference 952
8.59 KDChartLineDiagram.cpp File Reference 953
8.60 KDChartLineDiagram.h File Reference 954
8.61 KDChartMarkerAttributes.cpp File Reference 955
8.62 KDChartMarkerAttributes.h File Reference 956
8.63 KDChartMeasure.cpp File Reference 957
8.64 KDChartMeasure.h File Reference 958
8.65 KDChartPaintContext.cpp File Reference 959
8.66 KDChartPaintContext.h File Reference 960
8.67 KDChartPalette.cpp File Reference 961
8.68 KDChartPalette.h File Reference 963
8.69 KDChartPieAttributes.cpp File Reference 964
8.70 KDChartPieAttributes.h File Reference 965
8.71 KDChartPieDiagram.cpp File Reference 966
8.72 KDChartPieDiagram.h File Reference 968

8.73 KDChartPolarCoordinatePlane.cpp File Reference 969
8.74 KDChartPolarCoordinatePlane.h File Reference 970
8.75 KDChartPolarDiagram.cpp File Reference 971
8.76 KDChartPolarDiagram.h File Reference 972
8.77 KDChartPosition.cpp File Reference 973
8.78 KDChartPosition.h File Reference 976
8.79 KDChartRelativePosition.cpp File Reference 977
8.80 KDChartRelativePosition.h File Reference 978
8.81 KDChartRingDiagram.cpp File Reference 979
8.82 KDChartRingDiagram.h File Reference 980
8.83 KDChartSignalCompressor.cpp File Reference 981
8.84 KDChartSignalCompressor.h File Reference 982
8.85 KDChartTextArea.cpp File Reference 983
8.86 KDChartTextArea.h File Reference 984
8.87 KDChartTextAttributes.cpp File Reference 985
8.88 KDChartTextAttributes.h File Reference 986
8.89 KDChartTextLabelCache.cpp File Reference 987
8.90 KDChartTextLabelCache.h File Reference 989
8.91 KDChartThreeDBarAttributes.cpp File Reference 990
8.92 KDChartThreeDBarAttributes.h File Reference 991
8.93 KDChartThreeDLineAttributes.cpp File Reference 992
8.94 KDChartThreeDLineAttributes.h File Reference 993
8.95 KDChartThreeDPieAttributes.cpp File Reference 994
8.96 KDChartThreeDPieAttributes.h File Reference 995
8.97 KDChartWidget.cpp File Reference 996
8.98 KDChartWidget.h File Reference 998
8.99 KDChartZoomParameters.h File Reference 999
8.100 KDTextDocument.cpp File Reference 1000
8.101 KDTextDocument.h File Reference 1001

9 KD Chart 2 Page Documentation 1003

9.1 Deprecated List 1003

Chapter 1

KD Chart 2 Namespace Index

1.1 KD Chart 2 Namespace List

Here is a list of all namespaces with brief descriptions:

BackgroundAttributes 15
BarAttributes 16
DataValueAttributes 17
FrameAttributes 18
GridAttributes 19
KDChart () 20
LineAttributes 25
MarkerAttributes 26
PaintContext 27
Palette 28
RelativePosition 29
TextAttributes 30
Ui (PRIVATE_API_DOCU) 31

Chapter 2

KD Chart 2 Hierarchical Index

2.1 KD Chart 2 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

- KDChart::AbstractAreaBase . . . 45
  - KDChart::AbstractArea . . . 33
    - KDChart::AbstractAxis . . . 61
      - KDChart::CartesianAxis . . . 326
    - KDChart::AbstractCoordinatePlane . . . 115
      - KDChart::CartesianCoordinatePlane . . . 363
      - KDChart::PolarCoordinatePlane . . . 665
  - KDChart::AbstractAreaWidget . . . 51
    - KDChart::Legend . . . 496
  - KDChart::TextArea . . . 813
    - KDChart::HeaderFooter . . . 460
- KDChart::AbstractThreeDAttributes . . . 249
  - KDChart::ThreeDBarAttributes . . . 844
  - KDChart::ThreeDLineAttributes . . . 849
  - KDChart::ThreeDPieAttributes . . . 854
- KDChart::BackgroundAttributes . . . 274
- BackgroundAttributes::Private
- KDChart::BarAttributes . . . 277
- BarAttributes::Private
- KDChart::DataDimension . . . 427
- KDChart::DataValueAttributes . . . 440
- DataValueAttributes::Private
- KDChart::FrameAttributes . . . 452
- FrameAttributes::Private
- KDChart::GlobalMeasureScaling . . . 454
- KDChart::GridAttributes . . . 457
- GridAttributes::Private
- KDChart::KDPrivateModelIndex
- KDChart::LineAttributes . . . 533
- LineAttributes::Private
- KDChart::MarkerAttributes . . . 599
- MarkerAttributes::Private

- KDChart::Measure 608
- KDChart::PaintContext 615
- PaintContext::Private
- Palette::Private
- KDChart::PieAttributes 620
- KDChart::Position 736
- KDChart::PositionPoints 745
- PrerenderedElement 749
  - PrerenderedLabel 752
- QAbstractItemView 758
  - KDChart::AbstractDiagram 141
    - KDChart::AbstractCartesianDiagram 79
      - KDChart::BarDiagram 280
      - KDChart::LineDiagram 536
    - KDChart::AbstractPolarDiagram 214
      - KDChart::AbstractPieDiagram 177
        - KDChart::PieDiagram 623
        - KDChart::RingDiagram 771
      - KDChart::PolarDiagram 698
- QAbstractProxyModel 759
  - KDChart::AbstractProxyModel 247
    - KDChart::AttributesModel 252
      - KDChart::PrivateAttributesModel
- QFrame 760
  - KDChart::DatasetSelectorWidget 437
- QLayoutItem 761
  - KDChart::AbstractLayoutItem 173
    - KDChart::AbstractArea 33
    - KDChart::AutoSpacerLayoutItem 268
    - KDChart::HorizontalLineLayoutItem 479
    - KDChart::LineLayoutItem 587
    - KDChart::LineWithMarkerLayoutItem 593
    - KDChart::MarkerLayoutItem 602
    - KDChart::TextLayoutItem 834
      - KDChart::TextArea 813
    - KDChart::VerticalLineLayoutItem 858
- QObject 762
  - KDChart::AbstractArea 33
  - KDChart::DiagramObserver 448
  - KDChart::Palette 617
  - KDChart::SignalCompressor 811
  - KDChart::TextArea 813
  - KDChartEnums 484
- QSortFilterProxyModel 763
  - KDChart::DatasetProxyModel 430
- QTextDocument 764
  - KDTextDocument 494
- QWidget 765
  - KDChart::AbstractAreaWidget 51
  - KDChart::Chart 408

    - KDChart::Widget 863
- QWidgetItem
    - MyWidgetItem
- KDChart::RelativePosition 766
- RelativePosition::Private
- KDChart::TextAttributes 829
- TextAttributes::Private
- Widget
- KDChart::ZoomParameters 879

Chapter 3

KD Chart 2 Class Index

3.1 KD Chart 2 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

KDChart::AbstractArea (An area in the chart with a background, a frame, etc) 33
KDChart::AbstractAreaBase (Base class for AbstractArea and AbstractAreaWidget: An area in the chart with a background, a frame, etc) 45
KDChart::AbstractAreaWidget (An area in the chart with a background, a frame, etc) 51
KDChart::AbstractAxis (The base class for axes) 61
KDChart::AbstractCartesianDiagram (Base class for diagrams based on a cartesian coordianate system) 79
KDChart::AbstractCoordinatePlane 115
KDChart::AbstractDiagram (AbstractDiagram defines the interface for diagram classes) 141
KDChart::AbstractLayoutItem 173
KDChart::AbstractPieDiagram 177
KDChart::AbstractPolarDiagram 214
KDChart::AbstractProxyModel 247
KDChart::AbstractThreeDAttributes 249
KDChart::AttributesModel 252
KDChart::AutoSpacerLayoutItem 268
KDChart::BackgroundAttributes 274
KDChart::BarAttributes 277
KDChart::BarDiagram 280
KDChart::CartesianAxis (The class for cartesian axes) 326
KDChart::CartesianCoordinatePlane 363
KDChart::Chart (A chart with one or more diagrams) 408
KDChart::DataDimension (Helper class for one dimension of data, e.g. for the rows in a data model, or for the labels of an axis, or for the vertical lines in a grid) 427
KDChart::DatasetProxyModel (DatasetProxyModel takes a KDChart dataset configuration and translates it into a filtering proxy model) 430
KDChart::DatasetSelectorWidget 437
KDChart::DataValueAttributes (Diagram attributes dealing with data value labels) 440
KDChart::DiagramObserver (A DiagramObserver watches the associated diagram for changes and deletion and emits corresponsing signals) 448
KDChart::FrameAttributes 452
KDChart::GlobalMeasureScaling (Auxiliary class used by the KDChart::Measure and KDChart::Chart class) 454

KDChart::GridAttributes 457
KDChart::HeaderFooter 460
KDChart::HorizontalLineLayoutItem 479
KDChartEnums (Project global class providing some enums needed both by KDChartParams and by KDChartCustomBox) 484
KDTextDocument 494
KDChart::Legend (Legend defines the interface for the legend drawing class) 496
KDChart::LineAttributes 533
KDChart::LineDiagram 536
KDChart::LineLayoutItem 587
KDChart::LineWithMarkerLayoutItem 593
KDChart::MarkerAttributes 599
KDChart::MarkerLayoutItem 602
KDChart::Measure (Measure is used to specify all relative and/or absolute measures in KDChart, e.g. font sizes) 608
KDChart::PaintContext 615
KDChart::Palette (A Palette is a set of brushes (or colors) to be used for painting data sets) . . . 617
KDChart::PieAttributes 620
KDChart::PieDiagram 623
KDChart::PolarCoordinatePlane 665
KDChart::PolarDiagram 698
KDChart::Position (Defines a position, using compass terminology) 736
KDChart::PositionPoints 745
PrerenderedElement 749
PrerenderedLabel (CachedLabel is an internal KDChart class that simplifies creation and caching of cached text labels) 752
QAbstractItemView (Class only listed here to document inheritance of some KDChart classes) . 758
QAbstractProxyModel (Class only listed here to document inheritance of some KDChart classes) 759
QFrame (Class only listed here to document inheritance of some KDChart classes) 760
QLayoutItem 761
QObject (Class only listed here to document inheritance of some KDChart classes) 762
QSortFilterProxyModel (Class only listed here to document inheritance of some KDChart classes)763
QTextDocument 764
QWidget (Class only listed here to document inheritance of some KDChart classes) 765
KDChart::RelativePosition (Defines relative position information: reference area, position in this area, horizontal / vertical padding, and rotating) 766
KDChart::RingDiagram 771
KDChart::SignalCompressor (SignalCompressor compresses signals where the same signal needs to be emitted by several pieces of the code, but only one of the signals should be received at the end) 811
KDChart::TextArea (A text area in the chart with a background, a frame, etc) 813
KDChart::TextAttributes (A set of text attributes) 829
KDChart::TextLayoutItem 834
KDChart::ThreeDBarAttributes 844
KDChart::ThreeDLineAttributes 849
KDChart::ThreeDPieAttributes 854
KDChart::VerticalLineLayoutItem 858
KDChart::Widget (The KD Chart widget for usage without Model/View) 863
KDChart::ZoomParameters 879

Chapter 4

KD Chart 2 File Index

4.1 KD Chart 2 File List

Here is a list of all files with brief descriptions:

KDChartAbstractArea.cpp . . . 881
KDChartAbstractArea.h . . . 885
KDChartAbstractAreaBase.cpp . . . 886
KDChartAbstractAreaBase.h . . . 887
KDChartAbstractAreaWidget.cpp . . . 888
KDChartAbstractAreaWidget.h . . . 889
KDChartAbstractAxis.cpp . . . 890
KDChartAbstractAxis.h . . . 891
KDChartAbstractCartesianDiagram.cpp . . . 892
KDChartAbstractCartesianDiagram.h . . . 893
KDChartAbstractCoordinatePlane.cpp . . . 894
KDChartAbstractCoordinatePlane.h . . . 895
KDChartAbstractDiagram.cpp . . . 896
KDChartAbstractDiagram.h . . . 897
KDChartAbstractPieDiagram.cpp . . . 898
KDChartAbstractPieDiagram.h . . . 899
KDChartAbstractPolarDiagram.cpp . . . 900
KDChartAbstractPolarDiagram.h . . . 901
KDChartAbstractProxyModel.cpp . . . 902
KDChartAbstractProxyModel.h . . . 903
KDChartAbstractThreeDAttributes.cpp . . . 904
KDChartAbstractThreeDAttributes.h . . . 905
KDChartAttributesModel.cpp . . . 906
KDChartAttributesModel.h . . . 907
KDChartBackgroundAttributes.cpp . . . 908
KDChartBackgroundAttributes.h . . . 909
KDChartBarAttributes.cpp . . . 910
KDChartBarAttributes.h . . . 911
KDChartBarDiagram.cpp . . . 912
KDChartBarDiagram.h . . . 913
KDChartCartesianAxis.cpp . . . 914
KDChartCartesianAxis.h . . . 916
KDChartCartesianCoordinatePlane.cpp . . . 917

KDChartCartesianCoordinatePlane.h 918
KDChartChart.cpp 919
KDChartChart.h (Declaring the class KDChart::Chart) 922
KDChartDatasetProxyModel.cpp 923
KDChartDatasetProxyModel.h 924
KDChartDatasetSelector.cpp 925
KDChartDatasetSelector.h 926
KDChartDataValueAttributes.cpp 927
KDChartDataValueAttributes.h (Declaring the class KDChart::DataValueAttributes) 929
KDChartDiagramObserver.cpp 931
KDChartDiagramObserver.h 932
KDChartEnums.h (Definition of global enums) 933
KDChartFrameAttributes.cpp 934
KDChartFrameAttributes.h 935
KDChartGlobal.h 936
KDChartGridAttributes.cpp 941
KDChartGridAttributes.h 942
KDChartHeaderFooter.cpp 943
KDChartHeaderFooter.h 944
KDChartLayoutItems.cpp 945
KDChartLayoutItems.h 948
KDChartLegend.cpp 949
KDChartLegend.h 950
KDChartLineAttributes.cpp 951
KDChartLineAttributes.h 952
KDChartLineDiagram.cpp 953
KDChartLineDiagram.h 954
KDChartMarkerAttributes.cpp 955
KDChartMarkerAttributes.h 956
KDChartMeasure.cpp 957
KDChartMeasure.h (Declaring the class KDChart::Measure) 958
KDChartPaintContext.cpp 959
KDChartPaintContext.h 960
KDChartPalette.cpp 961
KDChartPalette.h 963
KDChartPieAttributes.cpp 964
KDChartPieAttributes.h 965
KDChartPieDiagram.cpp 966
KDChartPieDiagram.h 968
KDChartPolarCoordinatePlane.cpp 969
KDChartPolarCoordinatePlane.h 970
KDChartPolarDiagram.cpp 971
KDChartPolarDiagram.h 972
KDChartPosition.cpp 973
KDChartPosition.h 976
KDChartRelativePosition.cpp 977
KDChartRelativePosition.h 978
KDChartRingDiagram.cpp 979
KDChartRingDiagram.h 980
KDChartSignalCompressor.cpp 981
KDChartSignalCompressor.h 982
KDChartTextArea.cpp 983
KDChartTextArea.h 984
KDChartTextAttributes.cpp 985

KDChartTextAttributes.h 986
KDChartTextLabelCache.cpp 987
KDChartTextLabelCache.h 989
KDChartThreeDBarAttributes.cpp 990
KDChartThreeDBarAttributes.h 991
KDChartThreeDLineAttributes.cpp 992
KDChartThreeDLineAttributes.h 993
KDChartThreeDPieAttributes.cpp 994
KDChartThreeDPieAttributes.h 995
KDChartWidget.cpp 996
KDChartWidget.h 998
KDChartZoomParameters.h 999
KDTextDocument.cpp 1000
KDTextDocument.h 1001

Chapter 5

KD Chart 2 Page Index

5.1 KD Chart 2 Related Pages

Here is a list of all related documentation pages:

Deprecated List 1003

Chapter 6

KD Chart 2 Namespace Documentation

6.1 BackgroundAttributes Namespace Reference

6.2 BarAttributes Namespace Reference

6.3 DataValueAttributes Namespace Reference

6.4 FrameAttributes Namespace Reference

6.5 GridAttributes Namespace Reference

6.6 KDChart Namespace Reference

6.6.1 Detailed Description

Classes

- class AbstractArea

  An area in the chart with a background, a frame, etc.

- class AbstractAreaBase

  Base class for AbstractArea and AbstractAreaWidget: An area in the chart with a background, a frame, etc.

- class AbstractAreaWidget

  An area in the chart with a background, a frame, etc.

- class AbstractAxis

  The base class for axes.

- class AbstractCartesianDiagram

  Base class for diagrams based on a cartesian coordianate system.

- class AbstractCoordinatePlane
- class AbstractDiagram

  AbstractDiagram defines the interface for diagram classes.

- class AbstractLayoutItem
- class AbstractPieDiagram
- class AbstractPolarDiagram
- class AbstractProxyModel
- class AbstractThreeDAttributes
- class AttributesModel
- class AutoSpacerLayoutItem
- class BackgroundAttributes
- class BarAttributes
- class BarDiagram
- class CartesianAxis

  The class for cartesian axes.

- class CartesianCoordinatePlane
- class Chart

  A chart with one or more diagrams.

- class DataDimension

  Helper class for one dimension of data, e.g. for the rows in a data model, or for the labels of an axis, or for the vertical lines in a grid.

- class DatasetProxyModel

  DatasetProxyModel takes a KDChart dataset configuration and translates it into a filtering proxy model.

- class DatasetSelectorWidget

- class DataValueAttributes

  Diagram attributes dealing with data value labels.

- class DiagramObserver

  A DiagramObserver watches the associated diagram for changes and deletion and emits corresponsing signals.

- class FrameAttributes
- class GlobalMeasureScaling

  Auxiliary class used by the KDChart::Measure and KDChart::Chart class.

- class GridAttributes
- class HeaderFooter
- class HorizontalLineLayoutItem
- class Legend

  Legend defines the interface for the legend drawing class.

- class LineAttributes
- class LineDiagram
- class LineLayoutItem
- class LineWithMarkerLayoutItem
- class MarkerAttributes
- class MarkerLayoutItem
- class Measure

  Measure is used to specify all relative and/or absolute measures in KDChart, e.g. font sizes.

- class PaintContext
- class Palette

  A Palette is a set of brushes (or colors) to be used for painting data sets.

- class PieAttributes
- class PieDiagram
- class PolarCoordinatePlane
- class PolarDiagram
- class Position

  Defines a position, using compass terminology.

- class PositionPoints
- class RelativePosition

  Defines relative position information: reference area, position in this area, horizontal / vertical padding, and rotating.

- class RingDiagram
- class SignalCompressor

  SignalCompressor compresses signals where the same signal needs to be emitted by several pieces of the code, but only one of the signals should be received at the end.

- class TextArea

  A text area in the chart with a background, a frame, etc.

- class TextAttributes

  A set of text attributes.

- class TextLayoutItem
- class ThreeDBarAttributes
- class ThreeDLineAttributes
- class ThreeDPieAttributes
- class VerticalLineLayoutItem
- class Widget

  The KD Chart widget for usage without Model/View.

- class ZoomParameters

Typedefs

- typedef QList< AbstractDiagram ∗ > AbstractDiagramList
- typedef QList< CartesianAxis ∗ > CartesianAxisList
- typedef QList< const AbstractDiagram ∗ > ConstAbstractDiagramList
- typedef QList< const AbstractDiagram ∗ > ConstDiagramList
- typedef QList< AbstractCoordinatePlane ∗ > CoordinatePlaneList
- typedef QList< DataDimension > DataDimensionsList
- typedef QVector< int > DatasetDescriptionVector
- typedef QList< AbstractDiagram ∗ > DiagramList
- typedef QList< HeaderFooter ∗ > HeaderFooterList
- typedef QList< Legend ∗ > LegendList

Enumerations

- enum DisplayRoles {
  DatasetPenRole = 0x0A79EF95,
  DatasetBrushRole,
  DataValueLabelAttributesRole,
  ThreeDAttributesRole,
  LineAttributesRole,
  ThreeDLineAttributesRole,
  BarAttributesRole,
  ThreeDBarAttributesRole,
  PieAttributesRole,
  ThreeDPieAttributesRole,
  DataHiddenRole }

6.6.2 Typedef Documentation

6.6.2.1 typedef QList<AbstractDiagram∗> KDChart::AbstractDiagramList

Definition at line 606 of file KDChartAbstractDiagram.h.

Referenced by KDChart::AbstractCoordinatePlane::diagrams(), KDChart::PolarCoordinatePlane::paint(), and KDChart::CartesianCoordinatePlane::paint().

6.6.2.2 typedef QList<CartesianAxis∗> KDChart::CartesianAxisList

Definition at line 135 of file KDChartCartesianAxis.h.

Referenced by KDChart::AbstractCartesianDiagram::axes().

6.6.2.3 typedef QList<const AbstractDiagram∗> KDChart::ConstAbstractDiagramList

Definition at line 607 of file KDChartAbstractDiagram.h.

Referenced by KDChart::AbstractCoordinatePlane::diagrams().

6.6.2.4 typedef QList<const AbstractDiagram∗> KDChart::ConstDiagramList

Definition at line 43 of file KDChartLegend.h.

Referenced by KDChart::Legend::constDiagrams().

6.6.2.5 typedef QList<AbstractCoordinatePlane∗> KDChart::CoordinatePlaneList

Definition at line 50 of file KDChartChart.h.

Referenced by KDChart::Chart::coordinatePlanes().

6.6.2.6 typedef QList<DataDimension> KDChart::DataDimensionsList

Definition at line 42 of file KDChartAbstractCoordinatePlane.h.

Referenced by KDChart::PolarCoordinatePlane::getDataDimensionsList(), KDChart::CartesianCoordinatePlane::getDataDimensionsList(), KDChart::AbstractCoordinatePlane::gridDimensionsList(), KDChart::CartesianCoordinatePlane::layoutDiagrams(), and KDChart::CartesianAxis::paintCtx().

6.6.2.7 typedef QVector<int> KDChart::DatasetDescriptionVector

Definition at line 38 of file KDChartDatasetProxyModel.h.

Referenced by KDChart::DatasetProxyModel::setDatasetColumnDescriptionVector(), KDChart::DatasetProxyModel::setDatasetDescriptionVectors(), and KDChart::DatasetProxyModel::setDatasetRowDescriptionVector().

6.6.2.8 typedef QList<AbstractDiagram∗> KDChart::DiagramList

Definition at line 42 of file KDChartLegend.h.

Referenced by KDChart::Legend::diagrams().

6.6.2.9 typedef QList<HeaderFooter∗> KDChart::HeaderFooterList

Definition at line 51 of file KDChartChart.h.

Referenced by KDChart::Chart::headerFooters().

6.6.2.10 typedef QList<Legend∗> KDChart::LegendList

Definition at line 52 of file KDChartChart.h.

Referenced by KDChart::Chart::legends().

6.6.3 Enumeration Type Documentation

6.6.3.1 enum KDChart::DisplayRoles

Enumeration values:

- ***DatasetPenRole***
- ***DatasetBrushRole***
- ***DataValueLabelAttributesRole***
- ***ThreeDAttributesRole***
- ***LineAttributesRole***
- ***ThreeDLineAttributesRole***
- ***BarAttributesRole***
- ***ThreeDBarAttributesRole***
- ***PieAttributesRole***
- ***ThreeDPieAttributesRole***
- ***DataHiddenRole***

Definition at line 245 of file KDChartGlobal.h.

```
245                    {
246   DatasetPenRole = 0x0A79EF95,
247   DatasetBrushRole,
248   DataValueLabelAttributesRole,
249   ThreeDAttributesRole,
250   LineAttributesRole,
251   ThreeDLineAttributesRole,
252   BarAttributesRole,
253   ThreeDBarAttributesRole,
254   PieAttributesRole,
255   ThreeDPieAttributesRole,
256   DataHiddenRole
257 };
```

6.7 LineAttributes Namespace Reference

6.8 MarkerAttributes Namespace Reference

6.9 PaintContext Namespace Reference

6.10 Palette Namespace Reference

6.11 RelativePosition Namespace Reference

6.12 TextAttributes Namespace Reference

6.13 Ui Namespace Reference

6.13.1 Detailed Description

PRIVATE_API_DOCU

(This class is used internally by DatasetSelectorWidget.)

Chapter 7

KD Chart 2 Class Documentation

7.1 KDChart::AbstractArea Class Reference

`#include <KDChartAbstractArea.h>`

Inheritance diagram for KDChart::AbstractArea:Collaboration diagram for KDChart::AbstractArea:

7.1.1 Detailed Description

An area in the chart with a background, a frame, etc.

AbstractArea is the base class for all non-widget chart elements that have a set of background attributes and frame attributes, such as coordinate planes or axes.

Note:

> This class inherits from AbstractAreaBase, AbstractLayoutItem, QObject. The reason for this tripple inheritance is that neither AbstractAreaBase nor AbstractLayoutItem are QObject.

Definition at line 54 of file KDChartAbstractArea.h.

Public Member Functions

- void alignToReferencePoint (const RelativePosition &position)
- BackgroundAttributes backgroundAttributes () const
- virtual int bottomOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart:AutoSpacerLayoutItem::sizeHint().

- bool compare (const AbstractAreaBase ∗other) const

  Returns true if both areas have the same settings.

- FrameAttributes frameAttributes () const
- void getFrameLeadings (int &left, int &top, int &right, int &bottom) const
- virtual int leftOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- virtual void paint (QPainter ∗)=0

- virtual void paintAll (QPainter &painter)

  Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

- virtual void paintBackground (QPainter &painter, const QRect &rectangle)
- virtual void paintCtx (PaintContext ∗context)

  Default impl: Paint the complete item using its layouted position and size.

- virtual void paintFrame (QPainter &painter, const QRect &rectangle)
- virtual void paintIntoRect (QPainter &painter, const QRect &rect)

  Draws the background and frame, then calls paint().

- QLayout ∗ parentLayout ()
- void removeFromParentLayout ()
- virtual int rightOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- void setBackgroundAttributes (const BackgroundAttributes &a)
- void setFrameAttributes (const FrameAttributes &a)
- void setParentLayout (QLayout ∗lay)
- virtual void setParentWidget (QWidget ∗widget)

  Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

- virtual void sizeHintChanged () const

  Report changed size hint: ask the parent widget to recalculate the layout.

- virtual int topOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- virtual ∼AbstractArea ()

Static Public Member Functions

- void paintBackgroundAttributes (QPainter &painter, const QRect &rectangle, const KDChart::BackgroundAttributes &attributes)
- void paintFrameAttributes (QPainter &painter, const QRect &rectangle, const KDChart::FrameAttributes &attributes)

Protected Member Functions

- AbstractArea ()
- virtual QRect areaGeometry () const
- QRect innerRect () const
- virtual void positionHasChanged ()

Protected Attributes

- Q_SIGNALS __pad0__: void positionChanged(AbstractArea ∗)
- QWidget ∗ mParent
- QLayout ∗ mParentLayout

7.1.2 Constructor & Destructor Documentation

7.1.2.1 AbstractArea::~AbstractArea () `[virtual]`

Definition at line 62 of file KDChartAbstractArea.cpp.

```
{
    // this bloc left empty intentionally
}
```

7.1.2.2 AbstractArea::AbstractArea () `[protected]`

Definition at line 54 of file KDChartAbstractArea.cpp.

```
    : QObject()
    , KDChart::AbstractAreaBase()
    , KDChart::AbstractLayoutItem()
{
    init();
}
```

7.1.3 Member Function Documentation

7.1.3.1 void AbstractAreaBase::alignToReferencePoint (const RelativePosition & *position*) `[inherited]`

Definition at line 90 of file KDChartAbstractAreaBase.cpp.

```
{
    Q_UNUSED( position );
    // PENDING(kalle) FIXME
    qWarning( "Sorry, not implemented: void AbstractAreaBase::alignToReferencePoint( const RelativePosi
}
```

7.1.3.2 QRect AbstractArea::areaGeometry () const `[protected, virtual]`

Implements KDChart::AbstractAreaBase.

Definition at line 150 of file KDChartAbstractArea.cpp.

Referenced by KDChart::CartesianCoordinatePlane::drawingArea(), KDChart::PolarCoordinatePlane::layoutDiagrams(), KDChart::CartesianAxis::paint(), paintAll(), and KDChart::CartesianAxis::paintCtx().

```
{
    return geometry();
}
```

7.1.3.3 BackgroundAttributes AbstractAreaBase::backgroundAttributes () const `[inherited]`

Definition at line 112 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by updateCommonBrush().

```
113 {
114     return d->backgroundAttributes;
115 }
```

7.1.3.4 int AbstractArea::bottomOverlap (bool *doNotRecalculate* = false) const `[virtual]`

This is called at layout time by KDChart:AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the bottom edge of the area.

Note:

The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 101 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
102 {
103     // Re-calculate the sizes,
104     // so we also get the amountOf..Overlap members set newly:
105     if( ! doNotRecalculate )
106         sizeHint();
107     return d->amountOfBottomOverlap;
108 }
```

7.1.3.5 bool AbstractAreaBase::compare (const AbstractAreaBase ∗ *other*) const `[inherited]`

Returns true if both areas have the same settings.

Definition at line 75 of file KDChartAbstractAreaBase.cpp.

```
76 {
77     if( other == this ) return true;
78     if( ! other ){
79         //qDebug() << "CartesianAxis::compare() cannot compare to Null pointer";
80         return false;
81     }
82     /*
83     qDebug() << "AbstractAreaBase:" << (frameAttributes() == other->frameAttributes())
84         << (backgroundAttributes() == other->backgroundAttributes()) << "\n";
85     */
86     return  (frameAttributes()      == other->frameAttributes()) &&
87             (backgroundAttributes() == other->backgroundAttributes());
88 }
```

7.1.3.6 FrameAttributes AbstractAreaBase::frameAttributes () const `[inherited]`

Definition at line 102 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone(), and updateCommonBrush().

```
103 {
104     return d->frameAttributes;
105 }
```

7.1.3.7 void AbstractAreaBase::getFrameLeadings (int & *left*, int & *top*, int & *right*, int & *bottom*) const `[inherited]`

Definition at line 204 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::AbstractAreaBase::innerRect(), and KDChart::AbstractAreaWidget::paintAll().

```
205 {
206     if( d && d->frameAttributes.isVisible() ){
207         const int padding = qMax( d->frameAttributes.padding(), 0 );
208         left   = padding;
209         top    = padding;
210         right  = padding;
211         bottom = padding;
212     }else{
213         left   = 0;
214         top    = 0;
215         right  = 0;
216         bottom = 0;
217     }
218 }
```

7.1.3.8 QRect AbstractAreaBase::innerRect () const `[protected, inherited]`

Definition at line 220 of file KDChartAbstractAreaBase.cpp.

References KDChart::AbstractAreaBase::areaGeometry(), and KDChart::AbstractAreaBase::getFrameLeadings().

Referenced by KDChart::TextArea::paintAll(), and paintAll().

```
221 {
222     int left;
223     int top;
224     int right;
225     int bottom;
226     getFrameLeadings( left, top, right, bottom );
227     return
228         QRect( QPoint(0,0), areaGeometry().size() )
229             .adjusted( left, top, -right, -bottom );
230 }
```

7.1.3.9 int AbstractArea::leftOverlap (bool *doNotRecalculate* = false) const `[virtual]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the left edge of the area.

Note:

The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 77 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
    // Re-calculate the sizes,
    // so we also get the amountOf..Overlap members set newly:
    if( ! doNotRecalculate )
        sizeHint();
    return d->amountOfLeftOverlap;
}
```

7.1.3.10 virtual void KDChart::AbstractLayoutItem::paint (QPainter ∗) `[pure virtual, inherited]`

Implemented in KDChart::CartesianAxis, KDChart::CartesianCoordinatePlane, KDChart::TextLayoutItem, KDChart::MarkerLayoutItem, KDChart::LineLayoutItem, KDChart::LineWithMarkerLayoutItem, KDChart::HorizontalLineLayoutItem, KDChart::VerticalLineLayoutItem, KDChart::AutoSpacerLayoutItem, and KDChart::PolarCoordinatePlane.

Referenced by KDChart::Legend::paint(), KDChart::AbstractLayoutItem::paintAll(), paintAll(), and KDChart::AbstractLayoutItem::paintCtx().

7.1.3.11 void AbstractArea::paintAll (QPainter & *painter*) `[virtual]`

Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

Reimplemented from KDChart::AbstractLayoutItem.

Definition at line 123 of file KDChartAbstractArea.cpp.

References areaGeometry(), d, KDChart::AbstractAreaBase::innerRect(), KDChart::AbstractLayoutItem::paint(), KDChart::AbstractAreaBase::paintBackground(), and KDChart::AbstractAreaBase::paintFrame().

Referenced by paintIntoRect().

```
{
    // Paint the background and frame
    const QRect overlappingArea( geometry().adjusted(
            -d->amountOfLeftOverlap,
            -d->amountOfTopOverlap,
            d->amountOfRightOverlap,
            d->amountOfBottomOverlap ) );
    paintBackground( painter, overlappingArea );
    paintFrame(      painter, overlappingArea );

    // temporarily adjust the widget size, to be sure all content gets calculated
    // to fit into the inner rectangle
    const QRect oldGeometry( areaGeometry()  );
    QRect inner( innerRect() );
    inner.moveTo(
        oldGeometry.left() + inner.left(),
        oldGeometry.top()  + inner.top() );
    const bool needAdjustGeometry = oldGeometry != inner;
    if( needAdjustGeometry )
        setGeometry( inner );
    paint( &painter );
    if( needAdjustGeometry )
        setGeometry( oldGeometry );
    //qDebug() << "AbstractAreaWidget::paintAll() done.";
}
```

7.1.3.12 void AbstractAreaBase::paintBackground (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 188 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintBackgroundAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and paintAll().

```
{
    Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintBackground()",
                "Private class was not initialized!" );
    paintBackgroundAttributes( painter, rect, d->backgroundAttributes );
}
```

7.1.3.13 void AbstractAreaBase::paintBackgroundAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::BackgroundAttributes & *attributes*) `[static, inherited]`

Definition at line 119 of file KDChartAbstractAreaBase.cpp.

References KDChart::BackgroundAttributes::brush(), KDChart::BackgroundAttributes::isVisible(), KDChart::BackgroundAttributes::pixmap(), and KDChart::BackgroundAttributes::pixmapMode().

Referenced by KDChart::AbstractAreaBase::paintBackground().

```
{
    if( !attributes.isVisible() ) return;

    /* first draw the brush (may contain a pixmap)*/
    if( Qt::NoBrush != attributes.brush().style() ) {
        KDChart::PainterSaver painterSaver( &painter );
        painter.setPen( Qt::NoPen );
        const QPointF newTopLeft( painter.deviceMatrix().map( rect.topLeft() ) );
        painter.setBrushOrigin( newTopLeft );
        painter.setBrush( attributes.brush() );
        painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
    }
    /* next draw the backPixmap over the brush */
    if( !attributes.pixmap().isNull() &&
        attributes.pixmapMode() != BackgroundAttributes::BackgroundPixmapModeNone ) {
        QPointF ol = rect.topLeft();
        if( BackgroundAttributes::BackgroundPixmapModeCentered == attributes.pixmapMode() )
        {
            ol.setX( rect.center().x() - attributes.pixmap().width() / 2 );
            ol.setY( rect.center().y() - attributes.pixmap().height()/ 2 );
            painter.drawPixmap( ol, attributes.pixmap() );
        } else {
            QMatrix m;
            double zW = (double)rect.width()  / (double)attributes.pixmap().width();
            double zH = (double)rect.height() / (double)attributes.pixmap().height();
            switch( attributes.pixmapMode() ) {
            case BackgroundAttributes::BackgroundPixmapModeScaled:
            {
                double z;
                z = qMin( zW, zH );
                m.scale( z, z );
            }
            break;
            case BackgroundAttributes::BackgroundPixmapModeStretched:
                m.scale( zW, zH );
                break;
            default:
                ; // Cannot happen, previously checked
            }
            QPixmap pm = attributes.pixmap().transformed( m );
            ol.setX( rect.center().x() - pm.width() / 2 );
            ol.setY( rect.center().y() - pm.height()/ 2 );
            painter.drawPixmap( ol, pm );
        }
    }
}
```

7.1.3.14 void KDChart::AbstractLayoutItem::paintCtx (PaintContext ∗ *context*) `[virtual, inherited]`

Default impl: Paint the complete item using its layouted position and size.

Reimplemented in KDChart::CartesianAxis.

Definition at line 77 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint(), and KDChart::PaintContext::painter().

```
{
    if( context )
        paint( context->painter() );
}
```

7.1.3.15 void AbstractAreaBase::paintFrame (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 196 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintFrameAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and paintAll().

```
{
    Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintFrame()",
                "Private class was not initialized!" );
    paintFrameAttributes( painter, rect, d->frameAttributes );
}
```

7.1.3.16 void AbstractAreaBase::paintFrameAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::FrameAttributes & *attributes*) `[static, inherited]`

Definition at line 169 of file KDChartAbstractAreaBase.cpp.

References KDChart::FrameAttributes::isVisible(), and KDChart::FrameAttributes::pen().

Referenced by KDChart::AbstractAreaBase::paintFrame().

```
171 {
172
173     if( !attributes.isVisible() ) return;
174
175     // Note: We set the brush to NoBrush explicitely here.
176     //       Otherwise we might get a filled rectangle, so any
177     //       previously drawn background would be overwritten by that area.
178
179     const QPen   oldPen(   painter.pen() );
180     const QBrush oldBrush( painter.brush() );
181     painter.setPen(   attributes.pen() );
182     painter.setBrush( Qt::NoBrush );
183     painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
184     painter.setBrush( oldBrush );
185     painter.setPen(   oldPen );
186 }
```

7.1.3.17 void AbstractArea::paintIntoRect (QPainter & *painter*, const QRect & *rect*) `[virtual]`

Draws the background and frame, then calls paint().

In most cases there is no need to overwrite this method in a derived class, but you would overwrite AbstractLayoutItem::paint() instead.

Definition at line 111 of file KDChartAbstractArea.cpp.

References paintAll().

```
112 {
113     const QRect oldGeometry( geometry() );
114     if( oldGeometry != rect )
115         setGeometry( rect );
116     painter.translate( rect.left(), rect.top() );
117     paintAll( painter );
118     painter.translate( -rect.left(), -rect.top() );
119     if( oldGeometry != rect )
120         setGeometry( oldGeometry );
121 }
```

7.1.3.18 QLayout∗ KDChart::AbstractLayoutItem::parentLayout () `[inherited]`

Definition at line 74 of file KDChartLayoutItems.h.

```
75         {
76             return mParentLayout;
77         }
```

7.1.3.19 void AbstractArea::positionHasChanged () `[protected, virtual]`

Reimplemented from KDChart::AbstractAreaBase.

Definition at line 155 of file KDChartAbstractArea.cpp.

```
156 {
157     emit positionChanged( this );
158 }
```

7.1.3.20 void KDChart::AbstractLayoutItem::removeFromParentLayout () `[inherited]`

Definition at line 78 of file KDChartLayoutItems.h.

Referenced by KDChart::Chart::takeCoordinatePlane().

```
        {
            if( mParentLayout ){
                if( widget() )
                    mParentLayout->removeWidget( widget() );
                else
                    mParentLayout->removeItem( this );
            }
        }
```

7.1.3.21 int AbstractArea::rightOverlap (bool *doNotRecalculate* = false) const `[virtual]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the right edge of the area.

Note:

The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 85 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
    // Re-calculate the sizes,
    // so we also get the amountOf..Overlap members set newly:
    if( ! doNotRecalculate )
        sizeHint();
    return d->amountOfRightOverlap;
}
```

7.1.3.22 void AbstractAreaBase::setBackgroundAttributes (const BackgroundAttributes & *a*) `[inherited]`

Definition at line 107 of file KDChartAbstractAreaBase.cpp.

References d.

```
{
    d->backgroundAttributes = a;
}
```

7.1.3.23 void AbstractAreaBase::setFrameAttributes (const FrameAttributes & *a*) `[inherited]`

Definition at line 97 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone().

```
{
    d->frameAttributes = a;
}
```

7.1.3.24 void KDChart::AbstractLayoutItem::setParentLayout (QLayout ∗ *lay*) `[inherited]`

Definition at line 70 of file KDChartLayoutItems.h.

```
        {
            mParentLayout = lay;
        }
```

7.1.3.25 void KDChart::AbstractLayoutItem::setParentWidget (QWidget ∗ *widget*) `[virtual, inherited]`

Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

Thus, you need to call setParentWidget on every item, that has a non-fixed size.

Definition at line 64 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::mParent.

Referenced by KDChart::Legend::buildLegend(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
{
    mParent = widget;
}
```

7.1.3.26 void KDChart::AbstractLayoutItem::sizeHintChanged () const `[virtual, inherited]`

Report changed size hint: ask the parent widget to recalculate the layout.

Definition at line 86 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextLayoutItem::sizeHint().

```
{
    // This is exactly like what QWidget::updateGeometry does.
//  qDebug("KDChart::AbstractLayoutItem::sizeHintChanged() called");
    if( mParent ) {
        if ( mParent->layout() )
            mParent->layout()->invalidate();
        else
            QApplication::postEvent( mParent, new QEvent( QEvent::LayoutRequest ) );
    }
}
```

7.1.3.27 int AbstractArea::topOverlap (bool *doNotRecalculate* = false) const `[virtual]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the top edge of the area.

Note:

> The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 93 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
    // Re-calculate the sizes,
    // so we also get the amountOf..Overlap members set newly:
    if( ! doNotRecalculate )
        sizeHint();
    return d->amountOfTopOverlap;
}
```

7.1.4 Member Data Documentation

7.1.4.1 Q_SIGNALS KDChart::AbstractArea::__pad0__ `[protected]`

Reimplemented in KDChart::AbstractCoordinatePlane.

Definition at line 141 of file KDChartAbstractArea.h.

7.1.4.2 QWidget∗ KDChart::AbstractLayoutItem::mParent `[protected, inherited]`

Definition at line 88 of file KDChartLayoutItems.h.

Referenced by KDChart::AbstractLayoutItem::setParentWidget().

7.1.4.3 QLayout∗ KDChart::AbstractLayoutItem::mParentLayout `[protected, inherited]`

Definition at line 89 of file KDChartLayoutItems.h.

The documentation for this class was generated from the following files:

- KDChartAbstractArea.h
- KDChartAbstractArea.cpp

7.2 KDChart::AbstractAreaBase Class Reference

`#include <KDChartAbstractAreaBase.h>`

Inheritance diagram for KDChart::AbstractAreaBase:

7.2.1 Detailed Description

Base class for AbstractArea and AbstractAreaWidget: An area in the chart with a background, a frame, etc.

AbstractAreaBase is the base class for all chart elements that have a set of background attributes and frame attributes, such as legends or axes.

Note:

Normally you should not use AbstractAreaBase directly, but derive your classes from AbstractArea or AbstractAreaWidget.
This classis not a QObject, so it is easier to inherit from it, if your are inheriting from a QObject too like AbstractAreaWidget does it.

See also:

AbstractArea, AbstractAreaWidget

Definition at line 69 of file KDChartAbstractAreaBase.h.

Public Member Functions

- void alignToReferencePoint (const RelativePosition &position)
- BackgroundAttributes backgroundAttributes () const
- bool compare (const AbstractAreaBase ∗other) const

  Returns true if both areas have the same settings.

- FrameAttributes frameAttributes () const
- void getFrameLeadings (int &left, int &top, int &right, int &bottom) const
- virtual void paintBackground (QPainter &painter, const QRect &rectangle)
- virtual void paintFrame (QPainter &painter, const QRect &rectangle)
- void setBackgroundAttributes (const BackgroundAttributes &a)
- void setFrameAttributes (const FrameAttributes &a)

Static Public Member Functions

- void paintBackgroundAttributes (QPainter &painter, const QRect &rectangle, const KDChart::BackgroundAttributes &attributes)
- void paintFrameAttributes (QPainter &painter, const QRect &rectangle, const KDChart::FrameAttributes &attributes)

Protected Member Functions

- AbstractAreaBase ()
- virtual QRect areaGeometry () const=0
- QRect innerRect () const
- virtual void positionHasChanged ()
- virtual ∼AbstractAreaBase ()

7.2.2 Constructor & Destructor Documentation

7.2.2.1 AbstractAreaBase::AbstractAreaBase () `[protected]`

Definition at line 57 of file KDChartAbstractAreaBase.cpp.

```
57                                      :
58      _d( new Private() )
59 {
60 }
```

7.2.2.2 AbstractAreaBase::∼AbstractAreaBase () `[protected, virtual]`

Definition at line 62 of file KDChartAbstractAreaBase.cpp.

```
63 {
64      delete _d; _d = 0;
65 }
```

7.2.3 Member Function Documentation

7.2.3.1 void AbstractAreaBase::alignToReferencePoint (const RelativePosition & *position*)

Definition at line 90 of file KDChartAbstractAreaBase.cpp.

```
91 {
92      Q_UNUSED( position );
93      // PENDING(kalle) FIXME
94      qWarning( "Sorry, not implemented: void AbstractAreaBase::alignToReferencePoint( const RelativePosi
95 }
```

7.2.3.2 virtual QRect KDChart::AbstractAreaBase::areaGeometry () const `[protected, pure virtual]`

Implemented in KDChart::AbstractArea, KDChart::AbstractAreaWidget, and KDChart::TextArea.

Referenced by innerRect().

7.2.3.3 BackgroundAttributes AbstractAreaBase::backgroundAttributes () const

Definition at line 112 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by updateCommonBrush().

```
113 {
114      return d->backgroundAttributes;
115 }
```

7.2.3.4 bool AbstractAreaBase::compare (const AbstractAreaBase ∗ *other*) const

Returns true if both areas have the same settings.

Definition at line 75 of file KDChartAbstractAreaBase.cpp.

```
{
    if( other == this ) return true;
    if( ! other ){
        //qDebug() << "CartesianAxis::compare() cannot compare to Null pointer";
        return false;
    }
    /*
    qDebug() << "AbstractAreaBase:" << (frameAttributes() == other->frameAttributes())
        << (backgroundAttributes() == other->backgroundAttributes()) << "\n";
    */
    return  (frameAttributes()      == other->frameAttributes()) &&
            (backgroundAttributes() == other->backgroundAttributes());
}
```

7.2.3.5 FrameAttributes AbstractAreaBase::frameAttributes () const

Definition at line 102 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone(), and updateCommonBrush().

```
{
    return d->frameAttributes;
}
```

7.2.3.6 void AbstractAreaBase::getFrameLeadings (int & *left*, int & *top*, int & *right*, int & *bottom*) const

Definition at line 204 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by innerRect(), and KDChart::AbstractAreaWidget::paintAll().

```
{
    if( d && d->frameAttributes.isVisible() ){
        const int padding = qMax( d->frameAttributes.padding(), 0 );
        left   = padding;
        top    = padding;
        right  = padding;
        bottom = padding;
    }else{
        left   = 0;
        top    = 0;
        right  = 0;
        bottom = 0;
    }
}
```

7.2.3.7 QRect AbstractAreaBase::innerRect () const `[protected]`

Definition at line 220 of file KDChartAbstractAreaBase.cpp.

References areaGeometry(), and getFrameLeadings().

Referenced by KDChart::TextArea::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    int left;
    int top;
    int right;
    int bottom;
    getFrameLeadings( left, top, right, bottom );
    return
        QRect( QPoint(0,0), areaGeometry().size() )
            .adjusted( left, top, -right, -bottom );
}
```

7.2.3.8 void AbstractAreaBase::paintBackground (QPainter & *painter*, const QRect & *rectangle*) `[virtual]`

Definition at line 188 of file KDChartAbstractAreaBase.cpp.

References d, and paintBackgroundAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintBackground()",
                "Private class was not initialized!" );
    paintBackgroundAttributes( painter, rect, d->backgroundAttributes );
}
```

7.2.3.9 void AbstractAreaBase::paintBackgroundAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::BackgroundAttributes & *attributes*) `[static]`

Definition at line 119 of file KDChartAbstractAreaBase.cpp.

References KDChart::BackgroundAttributes::brush(), KDChart::BackgroundAttributes::isVisible(), KDChart::BackgroundAttributes::pixmap(), and KDChart::BackgroundAttributes::pixmapMode().

Referenced by paintBackground().

```
{
    if( !attributes.isVisible() ) return;

    /* first draw the brush (may contain a pixmap)*/
    if( Qt::NoBrush != attributes.brush().style() ) {
        KDChart::PainterSaver painterSaver( &painter );
        painter.setPen( Qt::NoPen );
        const QPointF newTopLeft( painter.deviceMatrix().map( rect.topLeft() ) );
        painter.setBrushOrigin( newTopLeft );
        painter.setBrush( attributes.brush() );
        painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
    }
    /* next draw the backPixmap over the brush */
    if( !attributes.pixmap().isNull() &&
        attributes.pixmapMode() != BackgroundAttributes::BackgroundPixmapModeNone ) {
        QPointF ol = rect.topLeft();
        if( BackgroundAttributes::BackgroundPixmapModeCentered == attributes.pixmapMode() )
        {
            ol.setX( rect.center().x() - attributes.pixmap().width() / 2 );
            ol.setY( rect.center().y() - attributes.pixmap().height()/ 2 );
            painter.drawPixmap( ol, attributes.pixmap() );
        } else {
            QMatrix m;
            double zW = (double)rect.width()  / (double)attributes.pixmap().width();
            double zH = (double)rect.height() / (double)attributes.pixmap().height();
            switch( attributes.pixmapMode() ) {
            case BackgroundAttributes::BackgroundPixmapModeScaled:
            {
                double z;
                z = qMin( zW, zH );
                m.scale( z, z );
            }
            break;
            case BackgroundAttributes::BackgroundPixmapModeStretched:
                m.scale( zW, zH );
                break;
            default:
                ; // Cannot happen, previously checked
            }
            QPixmap pm = attributes.pixmap().transformed( m );
            ol.setX( rect.center().x() - pm.width() / 2 );
            ol.setY( rect.center().y() - pm.height()/ 2 );
            painter.drawPixmap( ol, pm );
        }
    }
}
```

7.2.3.10 void AbstractAreaBase::paintFrame (QPainter & *painter*, const QRect & *rectangle*) `[virtual]`

Definition at line 196 of file KDChartAbstractAreaBase.cpp.

References d, and paintFrameAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintFrame()",
                "Private class was not initialized!" );
    paintFrameAttributes( painter, rect, d->frameAttributes );
}
```

7.2.3.11 void AbstractAreaBase::paintFrameAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::FrameAttributes & *attributes*) `[static]`

Definition at line 169 of file KDChartAbstractAreaBase.cpp.

References KDChart::FrameAttributes::isVisible(), and KDChart::FrameAttributes::pen().

Referenced by paintFrame().

```
{

```

```
173     if( !attributes.isVisible() ) return;
174
175     // Note: We set the brush to NoBrush explicitely here.
176     //       Otherwise we might get a filled rectangle, so any
177     //       previously drawn background would be overwritten by that area.
178
179     const QPen   oldPen(   painter.pen() );
180     const QBrush oldBrush( painter.brush() );
181     painter.setPen(   attributes.pen() );
182     painter.setBrush( Qt::NoBrush );
183     painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
184     painter.setBrush( oldBrush );
185     painter.setPen(   oldPen );
186 }
```

7.2.3.12 void AbstractAreaBase::positionHasChanged () `[protected, virtual]`

Reimplemented in KDChart::AbstractArea, KDChart::AbstractAreaWidget, and KDChart::TextArea.

Definition at line 232 of file KDChartAbstractAreaBase.cpp.

```
233 {
234     // this bloc left empty intentionally
235 }
```

7.2.3.13 void AbstractAreaBase::setBackgroundAttributes (const BackgroundAttributes & *a*)

Definition at line 107 of file KDChartAbstractAreaBase.cpp.

References d.

```
108 {
109     d->backgroundAttributes = a;
110 }
```

7.2.3.14 void AbstractAreaBase::setFrameAttributes (const FrameAttributes & *a*)

Definition at line 97 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone().

```
98 {
99     d->frameAttributes = a;
100 }
```

The documentation for this class was generated from the following files:

- KDChartAbstractAreaBase.h
- KDChartAbstractAreaBase.cpp

7.3 KDChart::AbstractAreaWidget Class Reference

`#include <KDChartAbstractAreaWidget.h>`

Inheritance diagram for KDChart::AbstractAreaWidget:Collaboration diagram for KDChart::AbstractAreaWidget:

7.3.1 Detailed Description

An area in the chart with a background, a frame, etc.

AbstractAreaWidget is the base for all widget classes that have a set of background attributes and frame attributes, such as KDChart::Chart and KDChart::Legend.

Definition at line 51 of file KDChartAbstractAreaWidget.h.

Public Member Functions

- AbstractAreaWidget (QWidget ∗parent=0)
- void alignToReferencePoint (const RelativePosition &position)
- BackgroundAttributes backgroundAttributes () const
- bool compare (const AbstractAreaBase ∗other) const

  Returns true if both areas have the same settings.

- virtual void forceRebuild ()

  Call this to trigger an unconditional re-building of the widget's internals.

- FrameAttributes frameAttributes () const
- void getFrameLeadings (int &left, int &top, int &right, int &bottom) const
- virtual void needSizeHint ()

  Call this to trigger an conditional re-building of the widget's internals.

- virtual void paint (QPainter ∗painter)=0

  Overwrite this to paint the inner contents of your widget.

- void paintAll (QPainter &painter)

  Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

- virtual void paintBackground (QPainter &painter, const QRect &rectangle)
- virtual void paintEvent (QPaintEvent ∗event)

  Draws the background and frame, then calls paint().

- virtual void paintFrame (QPainter &painter, const QRect &rectangle)
- virtual void paintIntoRect (QPainter &painter, const QRect &rect)

  Draws the background and frame, then calls paint().

- virtual void resizeLayout (const QSize &)
- void setBackgroundAttributes (const BackgroundAttributes &a)
- void setFrameAttributes (const FrameAttributes &a)

Static Public Member Functions

- void paintBackgroundAttributes (QPainter &painter, const QRect &rectangle, const KDChart::BackgroundAttributes &attributes)
- void paintFrameAttributes (QPainter &painter, const QRect &rectangle, const KDChart::FrameAttributes &attributes)

Public Attributes

- Q_SIGNALS __pad0__: void positionChanged(AbstractAreaWidget ∗)

Protected Member Functions

- virtual QRect areaGeometry () const
- QRect innerRect () const
- virtual void positionHasChanged ()
- virtual ∼AbstractAreaWidget ()

7.3.2 Constructor & Destructor Documentation

7.3.2.1 AbstractAreaWidget::AbstractAreaWidget (QWidget ∗ *parent* = 0) `[explicit]`

Definition at line 69 of file KDChartAbstractAreaWidget.cpp.

```
    : QWidget( parent )
    , AbstractAreaBase( new Private() )
{
    init();
}
```

7.3.2.2 AbstractAreaWidget::∼AbstractAreaWidget () `[protected, virtual]`

Definition at line 76 of file KDChartAbstractAreaWidget.cpp.

```
{
    // this block left empty intentionally
}
```

7.3.3 Member Function Documentation

7.3.3.1 void AbstractAreaBase::alignToReferencePoint (const RelativePosition & *position*) `[inherited]`

Definition at line 90 of file KDChartAbstractAreaBase.cpp.

```
{
    Q_UNUSED( position );
    // PENDING(kalle) FIXME
    qWarning( "Sorry, not implemented: void AbstractAreaBase::alignToReferencePoint( const RelativePosi
}
```

7.3.3.2 QRect AbstractAreaWidget::areaGeometry () const `[protected, virtual]`

Implements KDChart::AbstractAreaBase.

Definition at line 186 of file KDChartAbstractAreaWidget.cpp.

```
{
    return geometry();
}
```

7.3.3.3 BackgroundAttributes AbstractAreaBase::backgroundAttributes () const `[inherited]`

Definition at line 112 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by updateCommonBrush().

```
{
    return d->backgroundAttributes;
}
```

7.3.3.4 bool AbstractAreaBase::compare (const AbstractAreaBase $*$ *other*) const `[inherited]`

Returns true if both areas have the same settings.

Definition at line 75 of file KDChartAbstractAreaBase.cpp.

```
{
    if( other == this ) return true;
    if( ! other ){
        //qDebug() << "CartesianAxis::compare() cannot compare to Null pointer";
        return false;
    }
    /*
    qDebug() << "AbstractAreaBase:" << (frameAttributes() == other->frameAttributes())
        << (backgroundAttributes() == other->backgroundAttributes()) << "\n";
    */
    return  (frameAttributes()      == other->frameAttributes()) &&
            (backgroundAttributes() == other->backgroundAttributes());
}
```

7.3.3.5 void AbstractAreaWidget::forceRebuild () `[virtual]`

Call this to trigger an unconditional re-building of the widget's internals.

Reimplemented in KDChart::Legend.

Definition at line 140 of file KDChartAbstractAreaWidget.cpp.

```
{
    //bloc left empty intentionally
}
```

7.3.3.6 FrameAttributes AbstractAreaBase::frameAttributes () const `[inherited]`

Definition at line 102 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone(), and updateCommonBrush().

```
{
    return d->frameAttributes;
}
```

7.3.3.7 void AbstractAreaBase::getFrameLeadings (int & *left*, int & *top*, int & *right*, int & *bottom*) const `[inherited]`

Definition at line 204 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::AbstractAreaBase::innerRect(), and paintAll().

```
{
    if( d && d->frameAttributes.isVisible() ){
        const int padding = qMax( d->frameAttributes.padding(), 0 );
        left   = padding;
        top    = padding;
        right  = padding;
        bottom = padding;
    }else{
        left   = 0;
        top    = 0;
        right  = 0;
        bottom = 0;
    }
}
```

7.3.3.8 QRect AbstractAreaBase::innerRect () const `[protected, inherited]`

Definition at line 220 of file KDChartAbstractAreaBase.cpp.

References KDChart::AbstractAreaBase::areaGeometry(), and KDChart::AbstractAreaBase::getFrameLeadings().

Referenced by KDChart::TextArea::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    int left;
    int top;
    int right;
    int bottom;
    getFrameLeadings( left, top, right, bottom );
    return
        QRect( QPoint(0,0), areaGeometry().size() )
            .adjusted( left, top, -right, -bottom );
}
```

7.3.3.9 void AbstractAreaWidget::needSizeHint () `[virtual]`

Call this to trigger an conditional re-building of the widget's internals.

e.g. AbstractAreaWidget call this, before calling layout()->setGeometry()

Reimplemented in KDChart::Legend.

Definition at line 86 of file KDChartAbstractAreaWidget.cpp.

```
{
    // this block left empty intentionally
}
```

7.3.3.10 virtual void KDChart::AbstractAreaWidget::paint (QPainter ∗ *painter*) `[pure virtual]`

Overwrite this to paint the inner contents of your widget.

Note:

When overriding this method, please let your widget draw itself at the top/left corner of the painter. You should call rect() (or width(), height(), resp.) to find the drawable area's size: While the paint() method is being executed the frame of the widget is outside of its rect(), so you can use all of rect() for your custom drawing!

See also:

paint, paintIntoRect

Implemented in KDChart::Legend.

Referenced by paintAll().

7.3.3.11 void AbstractAreaWidget::paintAll (QPainter & *painter*)

Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

Definition at line 145 of file KDChartAbstractAreaWidget.cpp.

References KDChart::AbstractAreaBase::getFrameLeadings(), paint(), KDChart::AbstractAreaBase::paintBackground(), and KDChart::AbstractAreaBase::paintFrame().

Referenced by paintEvent(), and paintIntoRect().

```
{
    //qDebug() << "AbstractAreaWidget::paintAll() called";

    // Paint the background and frame
    paintBackground( painter, QRect(QPoint(0, 0), size() ) );
    paintFrame(      painter, QRect(QPoint(0, 0), size() ) );

/*
    we do not call setContentsMargins() now,
    but we call resizeLayout() whenever the size or the frame has changed

    // adjust the widget’s content margins,
    // to be sure all content gets calculated
    // to fit into the inner rectangle
160     const QRect oldGeometry( areaGeometry()  );
161     const QRect inner( innerRect() );
162     //qDebug() << "areaGeometry():" << oldGeometry
163     //         << "  contentsRect():" << contentsRect() << "  inner:" << inner;
164     if( contentsRect() != inner ){
165         //qDebug() << "old contentsRect():" << contentsRect() << "  new innerRect:" << inner;
166         setContentsMargins(
167             inner.left(),
168             inner.top(),
169             oldGeometry.width() -inner.width()-1,
170             oldGeometry.height()-inner.height()-1 );
171         //forceRebuild();
172     }
173 */
174     int left;
175     int top;
176     int right;
177     int bottom;
178     getFrameLeadings( left, top, right, bottom );
179     const QPoint translation( left, top );
180     painter.translate( translation );
181     paint( &painter );
182     painter.translate( -translation.x(), -translation.y() );
183      //qDebug() << "AbstractAreaWidget::paintAll() done.";
184 }
```

7.3.3.12 void AbstractAreaBase::paintBackground (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 188 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintBackgroundAttributes().

Referenced by KDChart::TextArea::paintAll(), paintAll(), and KDChart::AbstractArea::paintAll().

```
189 {
190     Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintBackground()",
191                 "Private class was not initialized!" );
192     paintBackgroundAttributes( painter, rect, d->backgroundAttributes );
193 }
```

7.3.3.13 void AbstractAreaBase::paintBackgroundAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::BackgroundAttributes & *attributes*) `[static, inherited]`

Definition at line 119 of file KDChartAbstractAreaBase.cpp.

References KDChart::BackgroundAttributes::brush(), KDChart::BackgroundAttributes::isVisible(), KDChart::BackgroundAttributes::pixmap(), and KDChart::BackgroundAttributes::pixmapMode().

Referenced by KDChart::AbstractAreaBase::paintBackground().

```
121 {
122     if( !attributes.isVisible() ) return;
123
124     /* first draw the brush (may contain a pixmap)*/
125     if( Qt::NoBrush != attributes.brush().style() ) {
126         KDChart::PainterSaver painterSaver( &painter );
127         painter.setPen( Qt::NoPen );
128         const QPointF newTopLeft( painter.deviceMatrix().map( rect.topLeft() ) );
129         painter.setBrushOrigin( newTopLeft );
130         painter.setBrush( attributes.brush() );
131         painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
132     }
133     /* next draw the backPixmap over the brush */
134     if( !attributes.pixmap().isNull() &&
135         attributes.pixmapMode() != BackgroundAttributes::BackgroundPixmapModeNone ) {
136         QPointF ol = rect.topLeft();
137         if( BackgroundAttributes::BackgroundPixmapModeCentered == attributes.pixmapMode() )
138         {
139             ol.setX( rect.center().x() - attributes.pixmap().width() / 2 );
140             ol.setY( rect.center().y() - attributes.pixmap().height()/ 2 );
141             painter.drawPixmap( ol, attributes.pixmap() );
142         } else {
143             QMatrix m;
144             double zW = (double)rect.width()  / (double)attributes.pixmap().width();
145             double zH = (double)rect.height() / (double)attributes.pixmap().height();
146             switch( attributes.pixmapMode() ) {
147             case BackgroundAttributes::BackgroundPixmapModeScaled:
148             {
149                 double z;
150                 z = qMin( zW, zH );
151                 m.scale( z, z );
152             }
153             break;
154             case BackgroundAttributes::BackgroundPixmapModeStretched:
155                 m.scale( zW, zH );
156                 break;
157             default:
158                 ; // Cannot happen, previously checked
159             }
160             QPixmap pm = attributes.pixmap().transformed( m );
161             ol.setX( rect.center().x() - pm.width() / 2 );
162             ol.setY( rect.center().y() - pm.height()/ 2 );
163             painter.drawPixmap( ol, pm );
164         }
165     }
166 }
```

7.3.3.14 void AbstractAreaWidget::paintEvent (QPaintEvent ∗ *event*) `[virtual]`

Draws the background and frame, then calls paint().

In most cases there is no need to overwrite this method in a derived class, but you would overwrite paint() instead.

See also:

paint

Definition at line 99 of file KDChartAbstractAreaWidget.cpp.

References d, and paintAll().

```
100 {
101     Q_UNUSED( event );
102     QPainter painter( this );
103     if( size() != d->currentLayoutSize ){
104         d->resizeLayout( this, size() );
105     }
106     paintAll( painter );
107 }
```

7.3.3.15 void AbstractAreaBase::paintFrame (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 196 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintFrameAttributes().

Referenced by KDChart::TextArea::paintAll(), paintAll(), and KDChart::AbstractArea::paintAll().

```
197 {
198     Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintFrame()",
199                 "Private class was not initialized!" );
200     paintFrameAttributes( painter, rect, d->frameAttributes );
201 }
```

7.3.3.16 void AbstractAreaBase::paintFrameAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::FrameAttributes & *attributes*) `[static, inherited]`

Definition at line 169 of file KDChartAbstractAreaBase.cpp.

References KDChart::FrameAttributes::isVisible(), and KDChart::FrameAttributes::pen().

Referenced by KDChart::AbstractAreaBase::paintFrame().

```
171 {
172
173     if( !attributes.isVisible() ) return;
174
175     // Note: We set the brush to NoBrush explicitely here.
176     //       Otherwise we might get a filled rectangle, so any
177     //       previously drawn background would be overwritten by that area.
178
179     const QPen   oldPen(   painter.pen() );
180     const QBrush oldBrush( painter.brush() );
181     painter.setPen(   attributes.pen() );
182     painter.setBrush( Qt::NoBrush );
183     painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
184     painter.setBrush( oldBrush );
185     painter.setPen(   oldPen );
186 }
```

7.3.3.17 void AbstractAreaWidget::paintIntoRect (QPainter & *painter*, const QRect & *rect*) `[virtual]`

Draws the background and frame, then calls paint().

In most cases there is no need to overwrite this method in a derived class, but you would overwrite paint() instead.

Definition at line 109 of file KDChartAbstractAreaWidget.cpp.

References d, and paintAll().

Referenced by KDChart::Chart::paint().

```
110 {
111     //qDebug() << "AbstractAreaWidget::paintIntoRect() called rect=" << rect;
112
113     if( rect.isEmpty() ) return;
```

```

    d->resizeLayout( this, rect.size() );

    const QPoint translation( rect.topLeft() );
    painter.translate( translation );
    paintAll( painter );
    painter.translate( -translation.x(), -translation.y() );

/*
    // make sure, the contents of the widget have been set up,
    // so we get a usefull geometry:
    needSizeHint();

    const QRect oldGeometry( layout()->geometry() );
    const QRect newGeo( QPoint(0,0), rect.size() );
    const bool mustChangeGeo = layout() && oldGeometry != newGeo;
    if( mustChangeGeo )
        layout()->setGeometry( newGeo );
    painter.translate( rect.left(), rect.top() );
    paintAll( painter );
    painter.translate( -rect.left(), -rect.top() );
    if( mustChangeGeo )
        layout()->setGeometry( oldGeometry );
*/
}
```

7.3.3.18 void AbstractAreaWidget::positionHasChanged () `[protected, virtual]`

Reimplemented from KDChart::AbstractAreaBase.

Definition at line 191 of file KDChartAbstractAreaWidget.cpp.

```
{
    emit positionChanged( this );
}
```

7.3.3.19 void AbstractAreaWidget::resizeLayout (const QSize &) `[virtual]`

Reimplemented in KDChart::Legend.

Definition at line 93 of file KDChartAbstractAreaWidget.cpp.

```
{
    Q_UNUSED( size );
    // this block left empty intentionally
}
```

7.3.3.20 void AbstractAreaBase::setBackgroundAttributes (const BackgroundAttributes & *a*) `[inherited]`

Definition at line 107 of file KDChartAbstractAreaBase.cpp.

References d.

```
{
    d->backgroundAttributes = a;
}
```

7.3.3.21 void AbstractAreaBase::setFrameAttributes (const FrameAttributes & *a*) `[inherited]`

Definition at line 97 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone().

```
98 {
99     d->frameAttributes = a;
100 }
```

7.3.4 Member Data Documentation

7.3.4.1 Q_SIGNALS KDChart::AbstractAreaWidget::__pad0__

Reimplemented in KDChart::Legend.

Definition at line 121 of file KDChartAbstractAreaWidget.h.

The documentation for this class was generated from the following files:

- KDChartAbstractAreaWidget.h
- KDChartAbstractAreaWidget.cpp

7.4 KDChart::AbstractAxis Class Reference

`#include <KDChartAbstractAxis.h>`

Inheritance diagram for KDChart::AbstractAxis:Collaboration diagram for KDChart::AbstractAxis:

7.4.1 Detailed Description

The base class for axes.

For being useful, axes need to be assigned to a diagram, see AbstractCartesianDiagram::addAxis and AbstractCartesianDiagram::takeAxis.

See also:

PolarAxis, AbstractCartesianDiagram

Definition at line 63 of file KDChartAbstractAxis.h.

Public Member Functions

- AbstractAxis (AbstractDiagram ∗diagram=0)
- void alignToReferencePoint (const RelativePosition &position)
- BackgroundAttributes backgroundAttributes () const
- virtual int bottomOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart:AutoSpacerLayoutItem::sizeHint().

- bool compare (const AbstractAreaBase ∗other) const

  Returns true if both areas have the same settings.

- bool compare (const AbstractAxis ∗other) const

  Returns true if both axes have the same settings.

- virtual void connectSignals ()

  Wireing the signal/slot connections.

- const AbstractCoordinatePlane ∗ coordinatePlane () const

  Convenience function, returns the coordinate plane, in which this axis is used.

- void createObserver (AbstractDiagram ∗diagram)
- virtual const QString customizedLabel (const QString &label) const

  Implement this method if you want to adjust axis labels before they are printed.

- void deleteObserver (AbstractDiagram ∗diagram)
- const AbstractDiagram ∗ diagram () const
- FrameAttributes frameAttributes () const
- virtual QRect geometry () const=0
- void getFrameLeadings (int &left, int &top, int &right, int &bottom) const
- QStringList labels () const

  Returns a list of strings, that are used as axis labels, as set via setLabels.

- virtual int leftOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- bool observedBy (AbstractDiagram ∗diagram) const
- virtual void paint (QPainter ∗)=0
- virtual void paintAll (QPainter &painter)

  Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

- virtual void paintBackground (QPainter &painter, const QRect &rectangle)
- virtual void paintCtx (PaintContext ∗context)

  Default impl: Paint the complete item using its layouted position and size.

- virtual void paintFrame (QPainter &painter, const QRect &rectangle)
- virtual void paintIntoRect (QPainter &painter, const QRect &rect)

  Draws the background and frame, then calls paint().

- QLayout ∗ parentLayout ()
- void removeFromParentLayout ()
- virtual int rightOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- void setBackgroundAttributes (const BackgroundAttributes &a)
- void setFrameAttributes (const FrameAttributes &a)
- virtual void setGeometry (const QRect &rect)=0
- void setLabels (const QStringList &list)

  Use this to specify your own set of strings, to be used as axis labels.

- void setParentLayout (QLayout ∗lay)
- virtual void setParentWidget (QWidget ∗widget)

  Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

- void setShortLabels (const QStringList &list)

  Use this to specify your own set of strings, to be used as axis labels, in case the normal labels are too long.

- void setTextAttributes (const TextAttributes &a)

  Use this to specify the text attributes to be used for axis labels.

- QStringList shortLabels () const

  Returns a list of strings, that are used as axis labels, as set via setShortLabels.

- virtual void sizeHintChanged () const

  Report changed size hint: ask the parent widget to recalculate the layout.

- TextAttributes textAttributes () const

  Returns the text attributes to be used for axis labels.

- virtual int topOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- virtual ∼AbstractAxis ()

Static Public Member Functions

- void paintBackgroundAttributes (QPainter &painter, const QRect &rectangle, const KDChart::BackgroundAttributes &attributes)
- void paintFrameAttributes (QPainter &painter, const QRect &rectangle, const KDChart::FrameAttributes &attributes)

Public Attributes

- public Q_SLOTS: void update()
- protected Q_SLOTS: virtual void delayedInit()

Protected Member Functions

- virtual QRect areaGeometry () const
- QRect innerRect () const
- virtual void positionHasChanged ()

Protected Attributes

- Q_SLOTS __pad0__: void positionChanged(AbstractArea ∗)
- QWidget ∗ mParent
- QLayout ∗ mParentLayout

7.4.2 Constructor & Destructor Documentation

7.4.2.1 AbstractAxis::AbstractAxis (AbstractDiagram ∗ *diagram* = 0) `[explicit]`

Definition at line 108 of file KDChartAbstractAxis.cpp.

```
    : AbstractArea( new Private( diagram, this ) )
{
    init();
    QTimer::singleShot(0, this, SLOT(delayedInit()));
}
```

7.4.2.2 AbstractAxis::∼AbstractAxis () `[virtual]`

Definition at line 115 of file KDChartAbstractAxis.cpp.

References d.

```
{
    d->mDiagram = 0;
    d->secondaryDiagrams.clear();
}
```

7.4.3 Member Function Documentation

7.4.3.1 void AbstractAreaBase::alignToReferencePoint (const RelativePosition & *position*) `[inherited]`

Definition at line 90 of file KDChartAbstractAreaBase.cpp.

```
{
    Q_UNUSED( position );
    // PENDING(kalle) FIXME
    qWarning( "Sorry, not implemented: void AbstractAreaBase::alignToReferencePoint( const RelativePosi
}
```

7.4.3.2 QRect AbstractArea::areaGeometry () const `[protected, virtual, inherited]`

Implements KDChart::AbstractAreaBase.

Definition at line 150 of file KDChartAbstractArea.cpp.

Referenced by KDChart::CartesianCoordinatePlane::drawingArea(), KDChart::PolarCoordinatePlane::layoutDiagrams(), KDChart::CartesianAxis::paint(), KDChart::AbstractArea::paintAll(), and KDChart::CartesianAxis::paintCtx().

```
{
    return geometry();
}
```

7.4.3.3 BackgroundAttributes AbstractAreaBase::backgroundAttributes () const `[inherited]`

Definition at line 112 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by updateCommonBrush().

```
{
    return d->backgroundAttributes;
}
```

7.4.3.4 int AbstractArea::bottomOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart:AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the bottom edge of the area.

Note:

> The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 101 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
    // Re-calculate the sizes,
    // so we also get the amountOf..Overlap members set newly:
    if( ! doNotRecalculate )
        sizeHint();
    return d->amountOfBottomOverlap;
}
```

7.4.3.5 bool AbstractAreaBase::compare (const AbstractAreaBase ∗ *other*) const `[inherited]`

Returns true if both areas have the same settings.

Definition at line 75 of file KDChartAbstractAreaBase.cpp.

```
{
    if( other == this ) return true;
    if( ! other ){
        //qDebug() << "CartesianAxis::compare() cannot compare to Null pointer";
        return false;
    }
    /*
    qDebug() << "AbstractAreaBase:" << (frameAttributes() == other->frameAttributes())
        << (backgroundAttributes() == other->backgroundAttributes()) << "\n";
    */
    return  (frameAttributes()      == other->frameAttributes()) &&
            (backgroundAttributes() == other->backgroundAttributes());
}
```

7.4.3.6 bool AbstractAxis::compare (const AbstractAxis ∗ *other*) const

Returns true if both axes have the same settings.

Definition at line 142 of file KDChartAbstractAxis.cpp.

```
{
    if( other == this ) return true;
    if( ! other ){
        //qDebug() << "CartesianAxis::compare() cannot compare to Null pointer";
        return false;
    }
    /*
    qDebug() << (textAttributes() == other->textAttributes());
    qDebug() << (labels()         == other->labels());
    qDebug() << (shortLabels()    == other->shortLabels());
    */
    return  ( static_cast<const AbstractAreaBase*>(this)->compare( other ) ) &&
            (textAttributes() == other->textAttributes()) &&
            (labels()         == other->labels()) &&
            (shortLabels()    == other->shortLabels());
}
```

7.4.3.7 void AbstractAxis::connectSignals () `[virtual]`

Wireing the signal/slot connections.

This method gets called automatically, each time, when you assign the axis to a diagram, either by passing a diagram∗ to the c'tor, or by calling the diagram's setAxis method, resp.

If overwriting this method in derived classes, make sure to call this base method AbstractAxis::connectSignals(), so your axis gets connected to the diagram's built-in signals.

See also:

AbstractCartesianDiagram::addAxis()

Definition at line 211 of file KDChartAbstractAxis.cpp.

References d.

Referenced by createObserver().

```
{
    if( d->observer ){
        connect( d->observer, SIGNAL( diagramDataChanged( AbstractDiagram *) ),
                this, SLOT( update() ) );
    }
}
```

7.4.3.8 const AbstractCoordinatePlane ∗ AbstractAxis::coordinatePlane () const

Convenience function, returns the coordinate plane, in which this axis is used.

If the axis is not used in a coordinate plane, the return value is Zero.

Definition at line 312 of file KDChartAbstractAxis.cpp.

References d.

```
{
    if( d->diagram() )
        return d->diagram()->coordinatePlane();
    return 0;
}
```

7.4.3.9 void AbstractAxis::createObserver (AbstractDiagram ∗ *diagram*)

Definition at line 177 of file KDChartAbstractAxis.cpp.

References connectSignals(), and d.

Referenced by KDChart::AbstractCartesianDiagram::addAxis().

```
{
    if( d->setDiagram( diagram ) )
        connectSignals();
}
```

7.4.3.10 const QString AbstractAxis::customizedLabel (const QString & *label*) const `[virtual]`

Implement this method if you want to adjust axis labels before they are printed.

KD Chart is calling this method immediately before drawing the text, this means: What you return here will be drawn without further modifications.

Parameters:

label The text of the label as KD Chart has calculated it automatically (or as it was taken from a QStringList provided by you, resp.)

Returns:

The text to be drawn. By default this is the same as `label`.

Definition at line 161 of file KDChartAbstractAxis.cpp.

Referenced by KDChart::CartesianAxis::maximumSize(), and KDChart::CartesianAxis::paintCtx().

```
162 {
163     return label;
164 }
```

7.4.3.11 void AbstractAxis::deleteObserver (AbstractDiagram ∗ *diagram*)

Definition at line 193 of file KDChartAbstractAxis.cpp.

References d.

Referenced by KDChart::AbstractCartesianDiagram::takeAxis(), and KDChart::AbstractCartesianDiagram::∼AbstractCartesianDiagram().

```
194 {
195     d->unsetDiagram( diagram );
196 }
```

7.4.3.12 const AbstractDiagram ∗ KDChart::AbstractAxis::diagram () const

Definition at line 319 of file KDChartAbstractAxis.cpp.

References d.

```
320 {
321     return d->diagram();
322 }
```

7.4.3.13 FrameAttributes AbstractAreaBase::frameAttributes () const `[inherited]`

Definition at line 102 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone(), and updateCommonBrush().

```
103 {
104     return d->frameAttributes;
105 }
```

7.4.3.14 virtual QRect KDChart::AbstractAxis::geometry () const `[pure virtual]`

Implemented in KDChart::CartesianAxis.

7.4.3.15 void AbstractAreaBase::getFrameLeadings (int & *left*, int & *top*, int & *right*, int & *bottom*) const `[inherited]`

Definition at line 204 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::AbstractAreaBase::innerRect(), and KDChart::AbstractAreaWidget::paintAll().

```
{
    if( d && d->frameAttributes.isVisible() ){
        const int padding = qMax( d->frameAttributes.padding(), 0 );
        left   = padding;
        top    = padding;
        right  = padding;
        bottom = padding;
    }else{
        left   = 0;
        top    = 0;
        right  = 0;
        bottom = 0;
    }
}
```

7.4.3.16 QRect AbstractAreaBase::innerRect () const `[protected, inherited]`

Definition at line 220 of file KDChartAbstractAreaBase.cpp.

References KDChart::AbstractAreaBase::areaGeometry(), and KDChart::AbstractAreaBase::getFrameLeadings().

Referenced by KDChart::TextArea::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    int left;
    int top;
    int right;
    int bottom;
    getFrameLeadings( left, top, right, bottom );
    return
        QRect( QPoint(0,0), areaGeometry().size() )
            .adjusted( left, top, -right, -bottom );
}
```

7.4.3.17 QStringList AbstractAxis::labels () const

Returns a list of strings, that are used as axis labels, as set via setLabels.

See also:

setLabels

Definition at line 273 of file KDChartAbstractAxis.cpp.

References d.

Referenced by KDChart::CartesianAxis::maximumSize(), and KDChart::CartesianAxis::paintCtx().

```
{
    return d->hardLabels;
}
```

7.4.3.18 int AbstractArea::leftOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the left edge of the area.

Note:

The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 77 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
   // Re-calculate the sizes,
   // so we also get the amountOf..Overlap members set newly:
   if( ! doNotRecalculate )
       sizeHint();
   return d->amountOfLeftOverlap;
}
```

7.4.3.19 bool KDChart::AbstractAxis::observedBy (AbstractDiagram ∗ *diagram*) const

Definition at line 324 of file KDChartAbstractAxis.cpp.

References d.

```
{
    return d->hasDiagram( diagram );
}
```

7.4.3.20 virtual void KDChart::AbstractLayoutItem::paint (QPainter ∗) `[pure virtual, inherited]`

Implemented in KDChart::CartesianAxis, KDChart::CartesianCoordinatePlane, KDChart::TextLayoutItem, KDChart::MarkerLayoutItem, KDChart::LineLayoutItem, KDChart::LineWithMarkerLayoutItem, KDChart::HorizontalLineLayoutItem, KDChart::VerticalLineLayoutItem, KDChart::AutoSpacerLayoutItem, and KDChart::PolarCoordinatePlane.

Referenced by KDChart::Legend::paint(), KDChart::AbstractLayoutItem::paintAll(), KDChart::AbstractArea::paintAll(), and KDChart::AbstractLayoutItem::paintCtx().

7.4.3.21 void AbstractArea::paintAll (QPainter & *painter*) `[virtual, inherited]`

Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

Reimplemented from KDChart::AbstractLayoutItem.

Definition at line 123 of file KDChartAbstractArea.cpp.

References KDChart::AbstractArea::areaGeometry(), d, KDChart::AbstractAreaBase::innerRect(), KDChart::AbstractLayoutItem::paint(), KDChart::AbstractAreaBase::paintBackground(), and KDChart::AbstractAreaBase::paintFrame().

Referenced by KDChart::AbstractArea::paintIntoRect().

```
124 {
125     // Paint the background and frame
126     const QRect overlappingArea( geometry().adjusted(
127             -d->amountOfLeftOverlap,
128             -d->amountOfTopOverlap,
129             d->amountOfRightOverlap,
130             d->amountOfBottomOverlap ) );
131     paintBackground( painter, overlappingArea );
132     paintFrame(      painter, overlappingArea );
133
134     // temporarily adjust the widget size, to be sure all content gets calculated
135     // to fit into the inner rectangle
136     const QRect oldGeometry( areaGeometry()  );
137     QRect inner( innerRect() );
138     inner.moveTo(
139         oldGeometry.left() + inner.left(),
140         oldGeometry.top()  + inner.top() );
141     const bool needAdjustGeometry = oldGeometry != inner;
142     if( needAdjustGeometry )
143         setGeometry( inner );
144     paint( &painter );
145     if( needAdjustGeometry )
146         setGeometry( oldGeometry );
147     //qDebug() << "AbstractAreaWidget::paintAll() done.";
148 }
```

7.4.3.22 void AbstractAreaBase::paintBackground (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 188 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintBackgroundAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
189 {
190     Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintBackground()",
191                 "Private class was not initialized!" );
192     paintBackgroundAttributes( painter, rect, d->backgroundAttributes );
193 }
```

7.4.3.23 void AbstractAreaBase::paintBackgroundAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::BackgroundAttributes & *attributes*) `[static, inherited]`

Definition at line 119 of file KDChartAbstractAreaBase.cpp.

References KDChart::BackgroundAttributes::brush(), KDChart::BackgroundAttributes::isVisible(), KDChart::BackgroundAttributes::pixmap(), and KDChart::BackgroundAttributes::pixmapMode().

Referenced by KDChart::AbstractAreaBase::paintBackground().

```
{
    if( !attributes.isVisible() ) return;

    /* first draw the brush (may contain a pixmap)*/
    if( Qt::NoBrush != attributes.brush().style() ) {
        KDChart::PainterSaver painterSaver( &painter );
        painter.setPen( Qt::NoPen );
        const QPointF newTopLeft( painter.deviceMatrix().map( rect.topLeft() ) );
        painter.setBrushOrigin( newTopLeft );
        painter.setBrush( attributes.brush() );
        painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
    }
    /* next draw the backPixmap over the brush */
    if( !attributes.pixmap().isNull() &&
        attributes.pixmapMode() != BackgroundAttributes::BackgroundPixmapModeNone ) {
        QPointF ol = rect.topLeft();
        if( BackgroundAttributes::BackgroundPixmapModeCentered == attributes.pixmapMode() )
        {
            ol.setX( rect.center().x() - attributes.pixmap().width() / 2 );
            ol.setY( rect.center().y() - attributes.pixmap().height()/ 2 );
            painter.drawPixmap( ol, attributes.pixmap() );
        } else {
            QMatrix m;
            double zW = (double)rect.width()  / (double)attributes.pixmap().width();
            double zH = (double)rect.height() / (double)attributes.pixmap().height();
            switch( attributes.pixmapMode() ) {
            case BackgroundAttributes::BackgroundPixmapModeScaled:
            {
                double z;
                z = qMin( zW, zH );
                m.scale( z, z );
            }
            break;
            case BackgroundAttributes::BackgroundPixmapModeStretched:
                m.scale( zW, zH );
                break;
            default:
                ; // Cannot happen, previously checked
            }
            QPixmap pm = attributes.pixmap().transformed( m );
            ol.setX( rect.center().x() - pm.width() / 2 );
            ol.setY( rect.center().y() - pm.height()/ 2 );
            painter.drawPixmap( ol, pm );
        }
    }
}
```

7.4.3.24 void KDChart::AbstractLayoutItem::paintCtx (PaintContext ∗ *context*) `[virtual, inherited]`

Default impl: Paint the complete item using its layouted position and size.

Reimplemented in KDChart::CartesianAxis.

Definition at line 77 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint(), and KDChart::PaintContext::painter().

```
{
    if( context )
        paint( context->painter() );
}
```

7.4.3.25 void AbstractAreaBase::paintFrame (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 196 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintFrameAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintFrame()",
                "Private class was not initialized!" );
    paintFrameAttributes( painter, rect, d->frameAttributes );
}
```

7.4.3.26 void AbstractAreaBase::paintFrameAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::FrameAttributes & *attributes*) `[static, inherited]`

Definition at line 169 of file KDChartAbstractAreaBase.cpp.

References KDChart::FrameAttributes::isVisible(), and KDChart::FrameAttributes::pen().

Referenced by KDChart::AbstractAreaBase::paintFrame().

```
{

    if( !attributes.isVisible() ) return;

    // Note: We set the brush to NoBrush explicitely here.
    //       Otherwise we might get a filled rectangle, so any
    //       previously drawn background would be overwritten by that area.

    const QPen   oldPen(   painter.pen() );
    const QBrush oldBrush( painter.brush() );
    painter.setPen(   attributes.pen() );
    painter.setBrush( Qt::NoBrush );
    painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
    painter.setBrush( oldBrush );
    painter.setPen(   oldPen );
}
```

7.4.3.27 void AbstractArea::paintIntoRect (QPainter & *painter*, const QRect & *rect*) `[virtual, inherited]`

Draws the background and frame, then calls paint().

In most cases there is no need to overwrite this method in a derived class, but you would overwrite AbstractLayoutItem::paint() instead.

Definition at line 111 of file KDChartAbstractArea.cpp.

References KDChart::AbstractArea::paintAll().

```
{
    const QRect oldGeometry( geometry() );
    if( oldGeometry != rect )
        setGeometry( rect );
    painter.translate( rect.left(), rect.top() );
    paintAll( painter );
    painter.translate( -rect.left(), -rect.top() );
    if( oldGeometry != rect )
        setGeometry( oldGeometry );
}
```

7.4.3.28 QLayout∗ KDChart::AbstractLayoutItem::parentLayout () [inherited]

Definition at line 74 of file KDChartLayoutItems.h.

```
        {
            return mParentLayout;
        }
```

7.4.3.29 void AbstractArea::positionHasChanged () [protected, virtual, inherited]

Reimplemented from KDChart::AbstractAreaBase.

Definition at line 155 of file KDChartAbstractArea.cpp.

```
{
    emit positionChanged( this );
}
```

7.4.3.30 void KDChart::AbstractLayoutItem::removeFromParentLayout () [inherited]

Definition at line 78 of file KDChartLayoutItems.h.

Referenced by KDChart::Chart::takeCoordinatePlane().

```
        {
            if( mParentLayout ){
                if( widget() )
                    mParentLayout->removeWidget( widget() );
                else
                    mParentLayout->removeItem( this );
            }
        }
```

7.4.3.31 int AbstractArea::rightOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the right edge of the area.

Note:

The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 85 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
    // Re-calculate the sizes,
    // so we also get the amountOf..Overlap members set newly:
    if( ! doNotRecalculate )
        sizeHint();
    return d->amountOfRightOverlap;
}
```

7.4.3.32 void AbstractAreaBase::setBackgroundAttributes (const BackgroundAttributes & *a*) `[inherited]`

Definition at line 107 of file KDChartAbstractAreaBase.cpp.

References d.

```
{
    d->backgroundAttributes = a;
}
```

7.4.3.33 void AbstractAreaBase::setFrameAttributes (const FrameAttributes & *a*) `[inherited]`

Definition at line 97 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone().

```
{
    d->frameAttributes = a;
}
```

7.4.3.34 virtual void KDChart::AbstractAxis::setGeometry (const QRect & *rect*) `[pure virtual]`

Implemented in KDChart::CartesianAxis.

7.4.3.35 void AbstractAxis::setLabels (const QStringList & *list*)

Use this to specify your own set of strings, to be used as axis labels.

Labels specified via setLabels take precedence: If a non-empty list is passed, KD Chart will use these strings as axis labels, instead of calculating them.

If you a smaller number of strings than the number of labels drawn at this axis, KD Chart will iterate over the list, repeating the strings, until all labels are drawn. As an example you could specify the seven days of the week as abscissa labels, which would be repeatedly used then.

By passing an empty QStringList you can reset the default behaviour.

See also:
labels, setShortLabels

Definition at line 263 of file KDChartAbstractAxis.cpp.

References d.

```
{
    d->hardLabels = list;
}
```

7.4.3.36 void KDChart::AbstractLayoutItem::setParentLayout (QLayout ∗ *lay*) `[inherited]`

Definition at line 70 of file KDChartLayoutItems.h.

```
        {
            mParentLayout = lay;
        }
```

7.4.3.37 void KDChart::AbstractLayoutItem::setParentWidget (QWidget ∗ *widget*) `[virtual, inherited]`

Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

Thus, you need to call setParentWidget on every item, that has a non-fixed size.

Definition at line 64 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::mParent.

Referenced by KDChart::Legend::buildLegend(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
{
    mParent = widget;
}
```

7.4.3.38 void AbstractAxis::setShortLabels (const QStringList & *list*)

Use this to specify your own set of strings, to be used as axis labels, in case the normal labels are too long.

Note:

Setting done via setShortLabels will be ignored, if you did not pass a non-empty string list via setLabels too!

By passing an empty QStringList you can reset the default behaviour.

See also:

shortLabels, setLabels

Definition at line 289 of file KDChartAbstractAxis.cpp.

References d.

```
{
    d->hardShortLabels = list;
}
```

7.4.3.39 void AbstractAxis::setTextAttributes (const TextAttributes & *a*)

Use this to specify the text attributes to be used for axis labels.

By default, the reference area will be set at painting time. It will be the then-valid coordinate plane's parent widget, so normally, it will be the KDChart::Chart. Thus the labels of all of your axes in all of your diagrams within that Chart will be drawn in same font size, by default.

See also:

textAttributes, setLabels

Definition at line 231 of file KDChartAbstractAxis.cpp.

References d.

```
{
    d->textAttributes = a;
}
```

7.4.3.40 QStringList AbstractAxis::shortLabels () const

Returns a list of strings, that are used as axis labels, as set via setShortLabels.

Note:

Setting done via setShortLabels will be ignored, if you did not pass a non-empty string list via setLabels too!

See also:

setShortLabels

Definition at line 302 of file KDChartAbstractAxis.cpp.

References d.

Referenced by KDChart::CartesianAxis::paintCtx().

```
{
    return d->hardShortLabels;
}
```

7.4.3.41 void KDChart::AbstractLayoutItem::sizeHintChanged () const `[virtual, inherited]`

Report changed size hint: ask the parent widget to recalculate the layout.

Definition at line 86 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextLayoutItem::sizeHint().

```
{
    // This is exactly like what QWidget::updateGeometry does.
//  qDebug("KDChart::AbstractLayoutItem::sizeHintChanged() called");
    if( mParent ) {
        if ( mParent->layout() )
            mParent->layout()->invalidate();
        else
            QApplication::postEvent( mParent, new QEvent( QEvent::LayoutRequest ) );
    }
}
```

7.4.3.42 TextAttributes AbstractAxis::textAttributes () const

Returns the text attributes to be used for axis labels.

See also:

setTextAttributes

Definition at line 241 of file KDChartAbstractAxis.cpp.

References d.

Referenced by KDChart::CartesianAxis::maximumSize(), KDChart::CartesianAxis::paintCtx(), and KDChart::CartesianAxis::titleTextAttributes().

```
{
    return d->textAttributes;
}
```

7.4.3.43 int AbstractArea::topOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the top edge of the area.

Note:

The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 93 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
    // Re-calculate the sizes,
    // so we also get the amountOf..Overlap members set newly:
    if( ! doNotRecalculate )
        sizeHint();
    return d->amountOfTopOverlap;
}
```

7.4.4 Member Data Documentation

7.4.4.1 Q_SIGNALS KDChart::AbstractArea::__pad0__ `[protected, inherited]`

Reimplemented in KDChart::AbstractCoordinatePlane.

Definition at line 141 of file KDChartAbstractArea.h.

7.4.4.2 QWidget∗ KDChart::AbstractLayoutItem::mParent `[protected, inherited]`

Definition at line 88 of file KDChartLayoutItems.h.

Referenced by KDChart::AbstractLayoutItem::setParentWidget().

7.4.4.3 QLayout∗ KDChart::AbstractLayoutItem::mParentLayout `[protected, inherited]`

Definition at line 89 of file KDChartLayoutItems.h.

7.4.4.4 public KDChart::AbstractAxis::Q_SLOTS

Definition at line 129 of file KDChartAbstractAxis.h.

7.4.4.5 protected KDChart::AbstractAxis::Q_SLOTS

Definition at line 126 of file KDChartAbstractAxis.h.

The documentation for this class was generated from the following files:

- KDChartAbstractAxis.h
- KDChartAbstractAxis.cpp

7.5 KDChart::AbstractCartesianDiagram Class Reference

`#include <KDChartAbstractCartesianDiagram.h>`

Inheritance diagram for KDChart::AbstractCartesianDiagram:Collaboration diagram for KDChart::AbstractCartesianDiagram:

7.5.1 Detailed Description

Base class for diagrams based on a cartesian coordianate system.

The AbstractCartesianDiagram interface adds those elements that are specific to diagrams based on a cartesian coordinate system to the basic AbstractDiagram interface.

Definition at line 45 of file KDChartAbstractCartesianDiagram.h.

Public Member Functions

- AbstractCartesianDiagram (QWidget ∗parent=0, CartesianCoordinatePlane ∗plane=0)
- virtual void addAxis (CartesianAxis ∗axis)

  Add the axis to the diagram.

- bool allowOverlappingDataValueTexts () const
- bool antiAliasing () const
- virtual AttributesModel ∗ attributesModel () const

  Returns the AttributesModel, that is used by this diagram.

- virtual KDChart::CartesianAxisList axes () const
- QBrush brush (const QModelIndex &index) const

  Retrieve the brush to be used, for painting the datapoint at the given index in the model.

- QBrush brush (int dataset) const

  Retrieve the brush to be used for the given dataset.

- QBrush brush () const

  Retrieve the brush to be used for painting datapoints globally.

- bool compare (const AbstractDiagram ∗other) const

  Returns true if both diagrams have the same settings.

- bool compare (const AbstractCartesianDiagram ∗other) const

  Returns true if both diagrams have the same settings.

- AbstractCoordinatePlane ∗ coordinatePlane () const

  The coordinate plane associated with the diagram.

- const QPair< QPointF, QPointF > dataBoundaries () const

  Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

- virtual void dataChanged (const QModelIndex &topLeft, const QModelIndex &bottomRight)

[reimplemented]

- QList< QBrush > datasetBrushes () const

  The set of dataset brushes currently used, for use in legends, etc.

- int datasetDimension () const

  The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

- QStringList datasetLabels () const

  The set of dataset labels currently displayed, for use in legends, etc.

- QList< MarkerAttributes > datasetMarkers () const

  The set of dataset markers currently used, for use in legends, etc.

- QList< QPen > datasetPens () const

  The set of dataset pens currently used, for use in legends, etc.

- DataValueAttributes dataValueAttributes (const QModelIndex &index) const

  Retrieve the DataValueAttributes for the given index.

- DataValueAttributes dataValueAttributes (int column) const

  Retrieve the DataValueAttributes for the given dataset.

- DataValueAttributes dataValueAttributes () const

  Retrieve the DataValueAttributes speficied globally.

- virtual void doItemsLayout ()

  [reimplemented]

- virtual int horizontalOffset () const

  [reimplemented]

- virtual QModelIndex indexAt (const QPoint &point) const

  [reimplemented]

- bool isHidden (const QModelIndex &index) const

  Retrieve the hidden status for the given index.

- bool isHidden (int column) const

  Retrieve the hidden status for the given dataset.

- bool isHidden () const

  Retrieve the hidden status speficied globally.

- virtual bool isIndexHidden (const QModelIndex &index) const

  [reimplemented]

- QStringList itemRowLabels () const

  The set of item row labels currently displayed, for use in Abscissa axes, etc.

- virtual void layoutPlanes ()
- virtual QModelIndex moveCursor (CursorAction cursorAction, Qt::KeyboardModifiers modifiers)

  [reimplemented]

- virtual const int numberOfAbscissaSegments () const=0
- virtual const int numberOfOrdinateSegments () const=0
- virtual void paint (PaintContext ∗paintContext)=0

  Draw the diagram contents to the rectangle and painter, that are passed in as part of the paint context.

- void paintDataValueText (QPainter ∗painter, const QModelIndex &index, const QPointF &pos, double value)
- void paintMarker (QPainter ∗painter, const QModelIndex &index, const QPointF &pos)
- virtual void paintMarker (QPainter ∗painter, const MarkerAttributes &markerAttributes, const QBrush &brush, const QPen &, const QPointF &point, const QSizeF &size)
- QPen pen (const QModelIndex &index) const

  Retrieve the pen to be used, for painting the datapoint at the given index in the model.

- QPen pen (int dataset) const

  Retrieve the pen to be used for the given dataset.

- QPen pen () const

  Retrieve the pen to be used for painting datapoints globally.

- bool percentMode () const
- virtual AbstractCartesianDiagram ∗ referenceDiagram () const
- virtual QPointF referenceDiagramOffset () const
- virtual void resize (const QSizeF &area)=0

  Called by the widget's sizeEvent.

- virtual void scrollTo (const QModelIndex &index, ScrollHint hint=EnsureVisible)

  [reimplemented]

- void setAllowOverlappingDataValueTexts (bool allow)

  Set whether data value labels are allowed to overlap.

- void setAntiAliasing (bool enabled)

  Set whether anti-aliasing is to be used while rendering this diagram.

- virtual void setAttributesModel (AttributesModel ∗model)

  Associate an AttributesModel with this diagram.

- void setBrush (const QBrush &brush)

  Set the brush to be used, for painting all datasets in the model.

- void setBrush (int dataset, const QBrush &brush)

  Set the brush to be used, for painting the given dataset.

- void setBrush (const QModelIndex &index, const QBrush &brush)

  Set the brush to be used, for painting the datapoint at the given index.

- virtual void setCoordinatePlane (AbstractCoordinatePlane ∗plane)

  Set the coordinate plane associated with the diagram.

- void setDatasetDimension (int dimension)

  Sets the dataset dimension of the diagram.

- void setDataValueAttributes (const DataValueAttributes &a)

  Set the DataValueAttributes for all datapoints in the model.

- void setDataValueAttributes (int dataset, const DataValueAttributes &a)

  Set the DataValueAttributes for the given dataset.

- void setDataValueAttributes (const QModelIndex &index, const DataValueAttributes &a)

  Set the DataValueAttributes for the given index.

- void setHidden (bool hidden)

  Hide (or unhide, resp.) all datapoints in the model.

- void setHidden (int column, bool hidden)

  Hide (or unhide, resp.) a dataset.

- void setHidden (const QModelIndex &index, bool hidden)

  Hide (or unhide, resp.) a data cell.

- virtual void setModel (QAbstractItemModel ∗model)

  Associate a model with the diagram.

- void setPen (const QPen &pen)

  Set the pen to be used, for painting all datasets in the model.

- void setPen (int dataset, const QPen &pen)

  Set the pen to be used, for painting the given dataset.

- void setPen (const QModelIndex &index, const QPen &pen)

  Set the pen to be used, for painting the datapoint at the given index.

- void setPercentMode (bool percent)
- virtual void setReferenceDiagram (AbstractCartesianDiagram ∗diagram, const QPointF &offset=QPointF())
- virtual void setRootIndex (const QModelIndex &idx)

  Set the root index in the model, where the diagram starts referencing data for display.

- virtual void setSelection (const QRect &rect, QItemSelectionModel::SelectionFlags command)

  [reimplemented]

- virtual void takeAxis (CartesianAxis ∗axis)

  Removes the axis from the diagram, without deleting it.

- void update () const

- void useDefaultColors ()

  Set the palette to be used, for painting datasets to the default palette.

- void useRainbowColors ()

  Set the palette to be used, for painting datasets to the rainbow palette.

- virtual bool usesExternalAttributesModel () const

  Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

- void useSubduedColors ()

  Set the palette to be used, for painting datasets to the subdued palette.

- virtual int verticalOffset () const

  [reimplemented]

- virtual QRect visualRect (const QModelIndex &index) const

  [reimplemented]

- virtual QRegion visualRegionForSelection (const QItemSelection &selection) const

  [reimplemented]

- virtual ∼AbstractCartesianDiagram ()

Protected Member Functions

- QModelIndex attributesModelRootIndex () const
- virtual const QPair< QPointF, QPointF > calculateDataBoundaries () const=0
- virtual bool checkInvariants (bool justReturnTheStatus=false) const
- QModelIndex columnToIndex (int column) const
- void dataHidden ()

  This signal is emitted, when the hidden status of at least one data cell was (un)set.

- void modelsChanged ()

  This signal is emitted, when either the model or the AttributesModel is replaced.

- virtual void paintDataValueTexts (QPainter ∗painter)
- virtual void paintMarkers (QPainter ∗painter)
- void propertiesChanged ()

  Emitted upon change of a property of the Diagram.

- void setAttributesModelRootIndex (const QModelIndex &)
- void setDataBoundariesDirty () const
- virtual double threeDItemDepth (int column) const=0
- virtual double threeDItemDepth (const QModelIndex &index) const=0
- double valueForCell (int row, int column) const

  Helper method, retrieving the data value (DisplayRole) for a given row and column.

Protected Attributes

- Q_SIGNALS __pad0__: void layoutChanged(AbstractDiagram∗)

7.5.2 Constructor & Destructor Documentation

7.5.2.1 AbstractCartesianDiagram::AbstractCartesianDiagram (QWidget ∗ *parent* = 0, CartesianCoordinatePlane ∗ *plane* = 0) `[explicit]`

Definition at line 76 of file KDChartAbstractCartesianDiagram.cpp.

```
    : AbstractDiagram ( new Private(), parent, plane )
{
}
```

7.5.2.2 KDChart::AbstractCartesianDiagram::∼AbstractCartesianDiagram () `[virtual]`

Definition at line 81 of file KDChartAbstractCartesianDiagram.cpp.

References d, and KDChart::AbstractAxis::deleteObserver().

```
{
    Q_FOREACH( CartesianAxis* axis, d->axesList ) {
        axis->deleteObserver( this );
    }
    d->axesList.clear();
}
```

7.5.3 Member Function Documentation

7.5.3.1 void AbstractCartesianDiagram::addAxis (CartesianAxis ∗ *axis*) `[virtual]`

Add the axis to the diagram.

The diagram takes ownership of the axis and will delete it.

To gain back ownership (e.g. for assigning the axis to another diagram) use the takeAxis method, before calling addAxis on the other diagram.

See also:
takeAxis

Definition at line 89 of file KDChartAbstractCartesianDiagram.cpp.

References KDChart::AbstractAxis::createObserver(), d, and layoutPlanes().

```
{
    if ( !d->axesList.contains( axis ) ) {
        d->axesList.append( axis );
        axis->createObserver( this );
        layoutPlanes();
    }
}
```

7.5.3.2 bool AbstractDiagram::allowOverlappingDataValueTexts () const `[inherited]`

Returns:

Whether data value labels are allowed to overlap.

Definition at line 446 of file KDChartAbstractDiagram.cpp.

References d.

```
450 {
```

7.5.3.3 bool AbstractDiagram::antiAliasing () const `[inherited]`

Returns:

Whether anti-aliasing is to be used for rendering this diagram.

Definition at line 457 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::paint().

```
461 {
```

7.5.3.4 AttributesModel ∗ AbstractDiagram::attributesModel () const `[virtual, inherited]`

Returns the AttributesModel, that is used by this diagram.

By default each diagram owns its own AttributesModel, which should never be deleted. Only if a user-supplied AttributesModel has been set does the pointer returned here not belong to the diagram.

Returns:

The AttributesModel associated with the diagram.

See also:

setAttributesModel

Definition at line 286 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::RingDiagram::paint(), KDChart::PolarDiagram::paint(), and KDChart::BarDiagram::setBarAttributes().

```
287 {
288     return d->attributesModel;
289 }
```

7.5.3.5 QModelIndex AbstractDiagram::attributesModelRootIndex () const `[protected, inherited]`

returns a QModelIndex pointing into the AttributesModel that corresponds to the root index of the diagram.

Definition at line 310 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::calculateDataBoundaries(), KDChart::BarDiagram::calculateDataBoundaries(), KDChart::LineDiagram::numberOfAbscissaSegments(), KDChart::BarDiagram::numberOfAbscissaSegments(), KDChart::LineDiagram::numberOfOrdinateSegments(), KDChart::BarDiagram::numberOfOrdinateSegments(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and KDChart::AbstractDiagram::valueForCell().

```
{
```

7.5.3.6 KDChart::CartesianAxisList AbstractCartesianDiagram::axes () const `[virtual]`

Definition at line 108 of file KDChartAbstractCartesianDiagram.cpp.

References KDChart::CartesianAxisList, and d.

```
{
    return d->axesList;
}
```

7.5.3.7 QBrush AbstractDiagram::brush (const QModelIndex & *index*) const `[inherited]`

Retrieve the brush to be used, for painting the datapoint at the given index in the model.

Parameters:

index The index of the datapoint in the model.

Returns:

The brush to use for painting.

Definition at line 816 of file KDChartAbstractDiagram.cpp.

```
                              :
QRect AbstractDiagram::visualRect(const QModelIndex &) const
```

7.5.3.8 QBrush AbstractDiagram::brush (int *dataset*) const `[inherited]`

Retrieve the brush to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the brush for.

Returns:

The brush to use for painting.

Definition at line 808 of file KDChartAbstractDiagram.cpp.

```
815 {
```

7.5.3.9 QBrush AbstractDiagram::brush () const `[inherited]`

Retrieve the brush to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:
The brush to use for painting.

Definition at line 802 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), and KDChart::AbstractDiagram::paintMarker().

```
807 {
```

7.5.3.10 virtual const QPair<QPointF, QPointF> KDChart::AbstractDiagram::calculateDataBoundaries () const `[protected, pure virtual, inherited]`

Implemented in KDChart::BarDiagram, KDChart::LineDiagram, KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

Referenced by KDChart::AbstractDiagram::dataBoundaries().

7.5.3.11 bool AbstractDiagram::checkInvariants (bool *justReturnTheStatus* = false) const `[protected, virtual, inherited]`

Definition at line 930 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane().

Referenced by KDChart::RingDiagram::calculateDataBoundaries(), KDChart::PolarDiagram::calculateDataBoundaries(), KDChart::PieDiagram::calculateDataBoundaries(), KDChart::LineDiagram::calculateDataBoundaries(), KDChart::BarDiagram::calculateDataBoundaries(), KDChart::RingDiagram::paint(), KDChart::PolarDiagram::paint(), KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and KDChart::AbstractDiagram::paintMarker().

```
930                             {
931         Q_ASSERT_X ( model(), "AbstractDiagram::checkInvariants()",
932                      "There is no usable model set, for the diagram." );
933
934         Q_ASSERT_X ( coordinatePlane(), "AbstractDiagram::checkInvariants()",
935                      "There is no usable coordinate plane set, for the diagram." );
936     }
937     return model() && coordinatePlane();
938 }
939
940 int AbstractDiagram::datasetDimension( ) const
```

7.5.3.12 QModelIndex AbstractDiagram::columnToIndex (int *column*) const `[protected, inherited]`

Definition at line 317 of file KDChartAbstractDiagram.cpp.

```
{
```

7.5.3.13 bool AbstractDiagram::compare (const AbstractDiagram ∗ *other*) const `[inherited]`

Returns true if both diagrams have the same settings.

Definition at line 135 of file KDChartAbstractDiagram.cpp.

```
{
    if( other == this ) return true;
    if( ! other ){
        //qDebug() << "AbstractDiagram::compare() cannot compare to Null pointer";
        return false;
    }
    /*
    qDebug() << "\n             AbstractDiagram::compare() QAbstractScrollArea:";
            // compare QAbstractScrollArea properties
    qDebug() <<
            ((horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
            (verticalScrollBarPolicy()    == other->verticalScrollBarPolicy()));
    qDebug() << "AbstractDiagram::compare() QFrame:";
            // compare QFrame properties
    qDebug() <<
            ((frameShadow() == other->frameShadow()) &&
            (frameShape()   == other->frameShape()) &&
            (frameWidth()   == other->frameWidth()) &&
            (lineWidth()    == other->lineWidth()) &&
            (midLineWidth() == other->midLineWidth()));
    qDebug() << "AbstractDiagram::compare() QAbstractItemView:";
            // compare QAbstractItemView properties
    qDebug() <<
            ((alternatingRowColors() == other->alternatingRowColors()) &&
            (hasAutoScroll()         == other->hasAutoScroll()) &&
#if QT_VERSION > 0x040199
            (dragDropMode()          == other->dragDropMode()) &&
            (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
            (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
            (verticalScrollMode()    == other->verticalScrollMode()) &&
#endif
            (dragEnabled()           == other->dragEnabled()) &&
            (editTriggers()          == other->editTriggers()) &&
            (iconSize()              == other->iconSize()) &&
            (selectionBehavior()     == other->selectionBehavior()) &&
            (selectionMode()         == other->selectionMode()) &&
            (showDropIndicator()     == other->showDropIndicator()) &&
            (tabKeyNavigation()      == other->tabKeyNavigation()) &&
            (textElideMode()         == other->textElideMode()));
    qDebug() << "AbstractDiagram::compare() AttributesModel: ";
            // compare all of the properties stored in the attributes model
    qDebug() << attributesModel()->compare( other->attributesModel() );
    qDebug() << "AbstractDiagram::compare() own:";
            // compare own properties
    qDebug() <<
            ((rootIndex().column()            == other->rootIndex().column()) &&
            (rootIndex().row()                == other->rootIndex().row()) &&
            (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
            (antiAliasing()                   == other->antiAliasing()) &&
            (percentMode()                    == other->percentMode()) &&
            (datasetDimension()               == other->datasetDimension()));
    */
    return  // compare QAbstractScrollArea properties
            (horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
            (verticalScrollBarPolicy()   == other->verticalScrollBarPolicy()) &&
            // compare QFrame properties
            (frameShadow()  == other->frameShadow()) &&
            (frameShape()   == other->frameShape()) &&
            (frameWidth()   == other->frameWidth()) &&
            (lineWidth()    == other->lineWidth()) &&
            (midLineWidth() == other->midLineWidth()) &&
            // compare QAbstractItemView properties
            (alternatingRowColors()  == other->alternatingRowColors()) &&
            (hasAutoScroll()         == other->hasAutoScroll()) &&
#if QT_VERSION > 0x040199
            (dragDropMode()          == other->dragDropMode()) &&
            (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
            (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
            (verticalScrollMode()    == other->verticalScrollMode()) &&
#endif
            (dragEnabled()           == other->dragEnabled()) &&
            (editTriggers()          == other->editTriggers()) &&
            (iconSize()              == other->iconSize()) &&
            (selectionBehavior()     == other->selectionBehavior()) &&
            (selectionMode()         == other->selectionMode()) &&
            (showDropIndicator()     == other->showDropIndicator()) &&
            (tabKeyNavigation()      == other->tabKeyNavigation()) &&
            (textElideMode()         == other->textElideMode()) &&
            // compare all of the properties stored in the attributes model
            attributesModel()->compare( other->attributesModel() ) &&
            // compare own properties
            (rootIndex().column()             == other->rootIndex().column()) &&
            (rootIndex().row()                == other->rootIndex().row()) &&
            (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
            (antiAliasing()                   == other->antiAliasing()) &&
            (percentMode()                    == other->percentMode()) &&
            (datasetDimension()               == other->datasetDimension());
}
```

7.5.3.14 bool AbstractCartesianDiagram::compare (const AbstractCartesianDiagram ∗ *other*) const

Returns true if both diagrams have the same settings.

Definition at line 52 of file KDChartAbstractCartesianDiagram.cpp.

```
{
    if( other == this ) return true;
    if( ! other ){
        //qDebug() << "AbstractCartesianDiagram::compare() cannot compare to Null pointer";
        return false;
    }
    /*
    qDebug() << "\n             AbstractCartesianDiagram::compare():";
            // compare own properties
    qDebug() <<
            ((referenceDiagram() == other->referenceDiagram()) &&
            ((! referenceDiagram()) || (referenceDiagramOffset() == other->referenceDiagramOffset()))));
    */
    return  // compare the base class
            ( static_cast<const AbstractDiagram*>(this)->compare( other ) ) &&
            // compare own properties
            (referenceDiagram() == other->referenceDiagram()) &&
            ((! referenceDiagram()) || (referenceDiagramOffset() == other->referenceDiagramOffset()));
}
```

7.5.3.15 AbstractCoordinatePlane ∗ AbstractDiagram::coordinatePlane () const `[inherited]`

The coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. By default this is a CartesianCoordinatePlane.

Returns:
The coordinate plane associated with the diagram.

Definition at line 226 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractDiagram::checkInvariants(), layoutPlanes(), KDChart::PolarDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), KDChart::AbstractPolarDiagram::polarCoordinatePlane(), and setCoordinatePlane().

```
{
    return d->plane;
}
```

7.5.3.16 const QPair< QPointF, QPointF > AbstractDiagram::dataBoundaries () const `[inherited]`

Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

This method returns a chached result of calculations done by calculateDataBoundaries. Classes derived from AbstractDiagram must implement the calculateDataBoundaries function, to specify their own way of calculating the data boundaries. If derived classes want to force recalculation of the data boundaries, they can call setDataBoundariesDirty()

Returned value is in diagram coordinates.

Definition at line 231 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::calculateDataBoundaries(), and d.

Referenced by KDChart::CartesianCoordinatePlane::getRawDataBoundingRectFromDiagrams(), KDChart::PolarCoordinatePlane::layoutDiagrams(), KDChart::LineDiagram::paint(), and KDChart::BarDiagram::paint().

```
{
    if( d->databoundariesDirty ){
        d->databoundaries = calculateDataBoundaries ();
        d->databoundariesDirty = false;
    }
    return d->databoundaries;
}
```

7.5.3.17 void AbstractDiagram::dataChanged (const QModelIndex & *topLeft*, const QModelIndex & *bottomRight*) `[virtual, inherited]`

[reimplemented]

Definition at line 338 of file KDChartAbstractDiagram.cpp.

References d.

```
338 {
339   // We are still too dumb to do intelligent updates...
340   d->databoundariesDirty = true;
341   scheduleDelayedItemsLayout();
342 }
343
344
```

7.5.3.18 void KDChart::AbstractDiagram::dataHidden () `[protected, inherited]`

This signal is emitted, when the hidden status of at least one data cell was (un)set.

7.5.3.19 QList< QBrush > AbstractDiagram::datasetBrushes () const `[inherited]`

The set of dataset brushes currently used, for use in legends, etc.

Note:

Cell-level override brushes, if set, take precedence over the dataset values, so you might need to check these too, in order to find the brush, that is used for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 894 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), KDChart::Legend::datasetCount(), and KDChart::Legend::setBrushesFromDiagram().

```
896                                                                                  {
897         QBrush brush = qVariantValue<QBrush>( attributesModel()->headerData( i, Qt::Vertical, DatasetE
898         ret << brush;
899     }
900
901     return ret;
902 }
903
904 QList<QPen> AbstractDiagram::datasetPens() const
```

7.5.3.20 int AbstractDiagram::datasetDimension () const `[inherited]`

The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

For each dimension it will expect one column of values in the model. If the dimensionality is 1, automatic values will be used for the abscissa.

For example a diagram with the default dimension of 1, will have one column per datapoint (the y values) and will use automatic values for the x axis (1, 2, 3, ... n). If the dimension is 2, the diagram will use the first, (and the third, fifth, etc) columns as X values, and the second, (and the fourth, sixth, etc) column as Y values.

Returns:

The dataset dimension of the diagram.

Definition at line 942 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::calculateDataBoundaries(), KDChart::LineDiagram::getCellValues(), KDChart::CartesianCoordinatePlane::getDataDimensionsList(), KDChart::LineDiagram::paint(), and KDChart::LineDiagram::setType().

```
946 {
```

7.5.3.21 QStringList AbstractDiagram::datasetLabels () const `[inherited]`

The set of dataset labels currently displayed, for use in legends, etc.

Returns:

The set of dataset labels currently displayed.

Definition at line 882 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), and KDChart::Legend::datasetCount().

```
883                                                  : " << attributesModel()->columnCount(attributesModel
884     const int columnCount = attributesModel()->columnCount(attributesModelRootIndex());
885     for( int i = datasetDimension()-1; i < columnCount; i += datasetDimension() ){
886         //qDebug() << "dataset label: " << attributesModel()->headerData( i, Qt::Horizontal, Qt::Displ
887         ret << attributesModel()->headerData( i, Qt::Horizontal, Qt::DisplayRole ).toString();
888     }
889     return ret;
890 }
891
892 QList<QBrush> AbstractDiagram::datasetBrushes() const
```

7.5.3.22 QList< MarkerAttributes > AbstractDiagram::datasetMarkers () const `[inherited]`

The set of dataset markers currently used, for use in legends, etc.

Note:

Cell-level override markers, if set, take precedence over the dataset values, so you might need to check these too, in order to find the marker, that is shown for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 917 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
919                                                                                   {
920         DataValueAttributes a =
921             qVariantValue<DataValueAttributes>( attributesModel()->headerData( i, Qt::Vertical, DataVa
922         const MarkerAttributes &ma = a.markerAttributes();
923         ret << ma;
924     }
925     return ret;
926 }
927
928 bool AbstractDiagram::checkInvariants( bool justReturnTheStatus ) const
```

7.5.3.23 QList< QPen > AbstractDiagram::datasetPens () const `[inherited]`

The set of dataset pens currently used, for use in legends, etc.

Note:

Cell-level override pens, if set, take precedence over the dataset values, so you might need to check these too, in order to find the pens, that is used for a single cell.

Returns:

The current set of dataset pens.

Definition at line 906 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
                                                                                   {
        QPen pen = qVariantValue<QPen>( attributesModel()->headerData( i, Qt::Vertical, DatasetPenRole
        ret << pen;
    }
    return ret;
}

QList<MarkerAttributes> AbstractDiagram::datasetMarkers() const
```

7.5.3.24 DataValueAttributes AbstractDiagram::dataValueAttributes (const QModelIndex & *index*) const `[inherited]`

Retrieve the DataValueAttributes for the given index.

This will fall back automatically to what was set at dataset or model level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the attributes for.

Returns:

The DataValueAttributes for the given index.

Definition at line 427 of file KDChartAbstractDiagram.cpp.

```
{
```

7.5.3.25 DataValueAttributes AbstractDiagram::dataValueAttributes (int *column*) const `[inherited]`

Retrieve the DataValueAttributes for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the attributes for.

Returns:

The DataValueAttributes for the given dataset.

Definition at line 420 of file KDChartAbstractDiagram.cpp.

```
426 {
```

7.5.3.26 DataValueAttributes AbstractDiagram::dataValueAttributes () const `[inherited]`

Retrieve the DataValueAttributes speficied globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:
The global DataValueAttributes.

Definition at line 414 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::AbstractDiagram::paintDataValueText(), and KDChart::AbstractDiagram::paintMarker().

```
419 {
```

7.5.3.27 void AbstractDiagram::doItemsLayout () `[virtual, inherited]`

[reimplemented]

Definition at line 329 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::AbstractDiagram::update().

```
329                     {
330         d->plane->layoutDiagrams();
331         update();
332     }
333     QAbstractItemView::doItemsLayout();
334 }
335
336 void AbstractDiagram::dataChanged( const QModelIndex &topLeft,
```

7.5.3.28 int AbstractDiagram::horizontalOffset () const `[virtual, inherited]`

[reimplemented]

Definition at line 839 of file KDChartAbstractDiagram.cpp.

```
841 { return 0; }
```

7.5.3.29 QModelIndex AbstractDiagram::indexAt (const QPoint & *point*) const `[virtual, inherited]`

[reimplemented]

Definition at line 833 of file KDChartAbstractDiagram.cpp.

```
835 { return QModelIndex(); }
```

7.5.3.30 bool AbstractDiagram::isHidden (const QModelIndex & *index*) const `[inherited]`

Retrieve the hidden status for the given index.

This will fall back automatically to what was set at dataset or diagram level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the hidden status for.

Returns:

The hidden status for the given index.

Definition at line 386 of file KDChartAbstractDiagram.cpp.

7.5.3.31 bool AbstractDiagram::isHidden (int *column*) const `[inherited]`

Retrieve the hidden status for the given dataset.

This will fall back automatically to what was set at diagram level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the hidden status for.

Returns:

The hidden status for the given dataset.

Definition at line 379 of file KDChartAbstractDiagram.cpp.

```
385 {
```

7.5.3.32 bool AbstractDiagram::isHidden () const `[inherited]`

Retrieve the hidden status speficied globally.

This will fall back automatically to the default settings (= not hidden), if there are no specific settings.

Returns:

The global hidden status.

Definition at line 373 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), KDChart::LineDiagram::paint(), and KDChart::LineDiagram::valueForCellTesting().

```
378 {
```

7.5.3.33 bool AbstractDiagram::isIndexHidden (const QModelIndex & *index*) const `[virtual, inherited]`

[reimplemented]

Definition at line 845 of file KDChartAbstractDiagram.cpp.

```
847 {}
```

7.5.3.34 QStringList AbstractDiagram::itemRowLabels () const `[inherited]`

The set of item row labels currently displayed, for use in Abscissa axes, etc.

Returns:

The set of item row labels currently displayed.

Definition at line 870 of file KDChartAbstractDiagram.cpp.

```
                                                 : " << attributesModel()->rowCount(attributesModelRoo
    const int rowCount = attributesModel()->rowCount(attributesModelRootIndex());
    for( int i = 0; i < rowCount; ++i ){
        //qDebug() << "item row label: " << attributesModel()->headerData( i, Qt::Vertical, Qt::Displa
        ret << attributesModel()->headerData( i, Qt::Vertical, Qt::DisplayRole ).toString();
    }
    return ret;
}

QStringList AbstractDiagram::datasetLabels() const
```

7.5.3.35 void KDChart::AbstractCartesianDiagram::layoutPlanes () `[virtual]`

Definition at line 113 of file KDChartAbstractCartesianDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane(), and KDChart::AbstractCoordinatePlane::layoutPlanes().

Referenced by addAxis(), and takeAxis().

```
{
    //qDebug() << "KDChart::AbstractCartesianDiagram::layoutPlanes()";
    AbstractCoordinatePlane* plane = coordinatePlane();
    if( plane ){
        plane->layoutPlanes();
        //qDebug() << "KDChart::AbstractCartesianDiagram::layoutPlanes() OK";
    }
}
```

7.5.3.36 void KDChart::AbstractDiagram::modelsChanged () `[protected, inherited]`

This signal is emitted, when either the model or the AttributesModel is replaced.

Referenced by KDChart::AbstractDiagram::setAttributesModel(), and KDChart::AbstractDiagram::setModel().

7.5.3.37 QModelIndex AbstractDiagram::moveCursor (CursorAction *cursorAction*, Qt::KeyboardModifiers *modifiers*) `[virtual, inherited]`

[reimplemented]

Definition at line 836 of file KDChartAbstractDiagram.cpp.

```
{ return 0; }
```

7.5.3.38 virtual const int KDChart::AbstractCartesianDiagram::numberOfAbscissaSegments () const `[pure virtual]`

Implemented in KDChart::BarDiagram, and KDChart::LineDiagram.

7.5.3.39 virtual const int KDChart::AbstractCartesianDiagram::numberOfOrdinateSegments () const `[pure virtual]`

Implemented in KDChart::BarDiagram, and KDChart::LineDiagram.

7.5.3.40 virtual void KDChart::AbstractDiagram::paint (PaintContext ∗ *paintContext*) `[pure virtual, inherited]`

Draw the diagram contents to the rectangle and painter, that are passed in as part of the paint context.

Parameters:

paintContext All information needed for painting.

Implemented in KDChart::BarDiagram, KDChart::LineDiagram, KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

7.5.3.41 void AbstractDiagram::paintDataValueText (QPainter ∗ *painter*, const QModelIndex & *index*, const QPointF & *pos*, double *value*) `[inherited]`

Definition at line 474 of file KDChartAbstractDiagram.cpp.

References KDChart::RelativePosition::alignment(), KDChart::TextAttributes::calculatedFont(), d, KDChart::DataValueAttributes::dataLabel(), KDChart::AbstractDiagram::dataValueAttributes(), KDChart::DataValueAttributes::decimalDigits(), KDChart::TextAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::TextAttributes::pen(), KDChart::DataValueAttributes::position(), KDChart::DataValueAttributes::prefix(), KDChart::TextAttributes::rotation(), KDChart::DataValueAttributes::showRepetitiveDataLabels(), KDChart::DataValueAttributes::suffix(), and KDChart::DataValueAttributes::textAttributes().

Referenced by KDChart::RingDiagram::paint(), and KDChart::PolarDiagram::paint().

```
{
    // paint one data series
    const DataValueAttributes a( dataValueAttributes(index) );
    if ( !a.isVisible() ) return;

    // handle decimal digits
    int decimalDigits = a.decimalDigits();
    int decimalPos = QString::number(  value ).indexOf( QLatin1Char( ’.’ ) );
    QString roundedValue;
    if ( a.dataLabel().isNull() ) {
        if ( decimalPos > 0 && value != 0 )
            roundedValue =  roundValues ( value, decimalPos, decimalDigits );
        else
            roundedValue = QString::number(  value );
    } else
        roundedValue = a.dataLabel();
        // handle prefix and suffix
    if ( !a.prefix().isNull() )
        roundedValue.prepend( a.prefix() );

    if ( !a.suffix().isNull() )
        roundedValue.append( a.suffix() );

    const TextAttributes ta( a.textAttributes() );
    // FIXME draw the non-text bits, background, etc
    if ( ta.isVisible() ) {

        QPointF pt( pos );
        /* for debugging:
        PainterSaver painterSaver( painter );
        painter->setPen( Qt::black );
        painter->drawLine( pos - QPointF( 1,1), pos + QPointF( 1,1) );
        painter->drawLine( pos - QPointF(-1,1), pos + QPointF(-1,1) );
        */

        // adjust the text start point position, if alignment is not Bottom/Left
        const RelativePosition relPos( a.position( value >= 0.0 ) );
        const Qt::Alignment alignBottomLeft = Qt::AlignBottom | Qt::AlignLeft;
        const QFont calculatedFont( ta.calculatedFont( d->plane, KDChartEnums::MeasureOrientationMinim
        //qDebug() << "calculatedFont’s point size:" << calculatedFont.pointSizeF();
        if( (relPos.alignment() & alignBottomLeft) != alignBottomLeft ){
            const QRectF boundRect(
                    d->cachedFontMetrics( calculatedFont, this )->boundingRect( roundedValue ) );
            if( relPos.alignment() & Qt::AlignRight )
                pt.rx() -= boundRect.width();
            else if( relPos.alignment() & Qt::AlignHCenter )
                pt.rx() -= 0.5 * boundRect.width();

            if( relPos.alignment() & Qt::AlignTop )
                pt.ry() += boundRect.height();
            else if( relPos.alignment() & Qt::AlignVCenter )
                pt.ry() += 0.5 * boundRect.height();
        }

        // FIXME draw the non-text bits, background, etc

        if ( a.showRepetitiveDataLabels() ||
             pos.x() <= d->lastX ||
             d->lastRoundedValue != roundedValue ) {
            d->lastRoundedValue = roundedValue;
            d->lastX = pos.x();

            PainterSaver painterSaver( painter );
            painter->setPen( ta.pen() );
            painter->setFont( calculatedFont );
            painter->translate( pt );
            painter->rotate( ta.rotation() );
            painter->drawText( QPointF(0, 0), roundedValue );
        }
    }
}


```

7.5.3.42 void AbstractDiagram::paintDataValueTexts (QPainter ∗ *painter*) `[protected, virtual, inherited]`

Definition at line 576 of file KDChartAbstractDiagram.cpp.

```
                                                                               {
       for ( int j=0; j< rowCount; ++j ) {
           const QModelIndex index = model()->index( j, i, rootIndex() );
           double value = model()->data( index ).toDouble();
           const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
```

```
584             paintDataValueText( painter, index, pos, value );
585         }
586     }
587 }
588
589
```

7.5.3.43 void AbstractDiagram::paintMarker (QPainter ∗ *painter*, const QModelIndex & *index*, const QPointF & *pos*) `[inherited]`

Definition at line 592 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::brush(), KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractDiagram::dataValueAttributes(), KDChart::MarkerAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::DataValueAttributes::markerAttributes(), KDChart::MarkerAttributes::markerColor(), KDChart::MarkerAttributes::markerSize(), KDChart::AbstractDiagram::paintMarker(), and KDChart::MarkerAttributes::pen().

```
593 {
594
595     if ( !checkInvariants() ) return;
596     DataValueAttributes a = dataValueAttributes(index);
597     if ( !a.isVisible() ) return;
598     const MarkerAttributes &ma = a.markerAttributes();
599     if ( !ma.isVisible() ) return;
600
601     PainterSaver painterSaver( painter );
602     QSizeF maSize( ma.markerSize() );
603     QBrush indexBrush( brush( index ) );
604     QPen indexPen( ma.pen() );
605     if ( ma.markerColor().isValid() )
606         indexBrush.setColor( ma.markerColor() );
607
608     paintMarker( painter, ma, indexBrush, indexPen, pos, maSize );
609 }
610
611
```

7.5.3.44 void AbstractDiagram::paintMarker (QPainter ∗ *painter*, const MarkerAttributes & *markerAttributes*, const QBrush & *brush*, const QPen &, const QPointF & *point*, const QSizeF & *size*) `[virtual, inherited]`

Definition at line 614 of file KDChartAbstractDiagram.cpp.

References KDChart::MarkerAttributes::markerStyle().

Referenced by KDChart::MarkerLayoutItem::paintIntoRect(), and KDChart::AbstractDiagram::paintMarker().

```
618 {
619
620     const QPen oldPen( painter->pen() );
621     // Pen is used to paint 4Pixels - 1 Pixel - Ring and FastCross types.
622     // make sure to use the brush color - see above in those cases.
623     const bool isFourPixels = (markerAttributes.markerStyle() == MarkerAttributes::Marker4Pixels);
624     if( isFourPixels || (markerAttributes.markerStyle() == MarkerAttributes::Marker1Pixel) ){
625         // for high-performance point charts with tiny point markers:
626         painter->setPen( QPen( brush.color().light() ) );
627         if( isFourPixels ){
```

```
            const qreal x = pos.x();
            const qreal y = pos.y();
            painter->drawLine( QPointF(x-1.0,y-1.0),
                               QPointF(x+1.0,y-1.0) );
            painter->drawLine( QPointF(x-1.0,y),
                               QPointF(x+1.0,y) );
            painter->drawLine( QPointF(x-1.0,y+1.0),
                               QPointF(x+1.0,y+1.0) );
        }
        painter->drawPoint( pos );
    }else{
        PainterSaver painterSaver( painter );
        // we only a solid line surrounding the markers
        QPen painterPen( pen );
        painterPen.setStyle( Qt::SolidLine );
        painter->setPen( painterPen );
        painter->setBrush( brush );
        painter->setRenderHint ( QPainter::Antialiasing );
        painter->translate( pos );
        switch ( markerAttributes.markerStyle() ) {
            case MarkerAttributes::MarkerCircle:
                painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                            maSize.height(), maSize.width()) );
                break;
            case MarkerAttributes::MarkerSquare:
                {
                    QRectF rect( 0 - maSize.width()/2, 0 - maSize.height()/2,
                                maSize.width(), maSize.height() );
                    painter->drawRect( rect );
                    painter->fillRect( rect, brush.color() );
                    break;
                }
            case MarkerAttributes::MarkerDiamond:
                {
                    QVector <QPointF > diamondPoints;
                    QPointF top, left, bottom, right;
                    top    = QPointF( 0, 0 - maSize.height()/2 );
                    left   = QPointF( 0 - maSize.width()/2, 0 );
                    bottom = QPointF( 0, maSize.height()/2 );
                    right  = QPointF( maSize.width()/2, 0 );
                    diamondPoints << top << left << bottom << right;
                    painter->drawPolygon( diamondPoints );
                    break;
                }
            // both handled on top of the method:
            case MarkerAttributes::Marker1Pixel:
            case MarkerAttributes::Marker4Pixels:
                    break;
            case MarkerAttributes::MarkerRing:
                {
                    painter->setPen( QPen( brush.color() ) );
                    painter->setBrush( Qt::NoBrush );
                    painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                                        maSize.height(), maSize.width()) );
                    break;
                }
            case MarkerAttributes::MarkerCross:
                {
                    QRectF rect( maSize.width()*-0.5, maSize.height()*-0.2,
                                 maSize.width(), maSize.height()*0.4 );
                    painter->drawRect( rect );
                    rect.setTopLeft(QPointF( maSize.width()*-0.2, maSize.height()*-0.5 ));
                    rect.setSize(QSizeF( maSize.width()*0.4, maSize.height() ));
                    painter->drawRect( rect );
                    break;
                }
            case MarkerAttributes::MarkerFastCross:
                {
                    QPointF left, right, top, bottom;
                    left  = QPointF( -maSize.width()/2, 0 );
                    right = QPointF( maSize.width()/2, 0 );
                    top   = QPointF( 0, -maSize.height()/2 );
                    bottom= QPointF( 0, maSize.height()/2 );
                    painter->setPen( QPen( brush.color() ) );
                    painter->drawLine( left, right );
                    painter->drawLine(  top, bottom );
                    break;
                }
            default:
                Q_ASSERT_X ( false, "paintMarkers()",
                            "Type item does not match a defined Marker Type." );
        }
    }
    painter->setPen( oldPen );
}

void AbstractDiagram::paintMarkers( QPainter* painter )
```

7.5.3.45 void AbstractDiagram::paintMarkers (QPainter ∗ *painter*) `[protected, virtual, inherited]`

Definition at line 716 of file KDChartAbstractDiagram.cpp.

```
                                                                                {
       for ( int j=0; j< rowCount; ++j ) {
           const QModelIndex index = model()->index( j, i, rootIndex() );
           double value = model()->data( index ).toDouble();
           const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
           paintMarker( painter, index, pos );
       }
    }
}


```

7.5.3.46 QPen AbstractDiagram::pen (const QModelIndex & *index*) const `[inherited]`

Retrieve the pen to be used, for painting the datapoint at the given index in the model.

Parameters:

index The index of the datapoint in the model.

Returns:

The pen to use for painting.

Definition at line 770 of file KDChartAbstractDiagram.cpp.

```
{
```

7.5.3.47 QPen AbstractDiagram::pen (int *dataset*) const `[inherited]`

Retrieve the pen to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the pen for.

Returns:

The pen to use for painting.

Definition at line 762 of file KDChartAbstractDiagram.cpp.

```
{
```

7.5.3.48 QPen AbstractDiagram::pen () const `[inherited]`

Retrieve the pen to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The pen to use for painting.

Definition at line 756 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::PieDiagram::paint(), and KDChart::LineDiagram::paint().

```
{
```

7.5.3.49 bool AbstractDiagram::percentMode () const `[inherited]`

Definition at line 468 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::CartesianCoordinatePlane::getDataDimensionsList().

7.5.3.50 void KDChart::AbstractDiagram::propertiesChanged () `[protected, inherited]`

Emitted upon change of a property of the Diagram.

Referenced by KDChart::LineDiagram::resetLineAttributes(), KDChart::AbstractDiagram::setDataValueAttributes(), KDChart::LineDiagram::setLineAttributes(), KDChart::LineDiagram::setThreeDLineAttributes(), and KDChart::LineDiagram::setType().

7.5.3.51 AbstractCartesianDiagram ∗ AbstractCartesianDiagram::referenceDiagram () const `[virtual]`

Definition at line 146 of file KDChartAbstractCartesianDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::paint(), and referenceDiagramIsBarDiagram().

```
{
    return d->referenceDiagram;
}
```

7.5.3.52 QPointF AbstractCartesianDiagram::referenceDiagramOffset () const `[virtual]`

Definition at line 151 of file KDChartAbstractCartesianDiagram.cpp.

References d.

```
152 {
153     return d->referenceDiagramOffset;
154 }
```

7.5.3.53 virtual void KDChart::AbstractDiagram::resize (const QSizeF & *area*) `[pure virtual, inherited]`

Called by the widget's sizeEvent.

Adjust all internal structures, that are calculated, dependending on the size of the widget.

Parameters:

area

Implemented in KDChart::BarDiagram, KDChart::LineDiagram, KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

7.5.3.54 void AbstractDiagram::scrollTo (const QModelIndex & *index*, ScrollHint *hint* = EnsureVisible) `[virtual, inherited]`

[reimplemented]

Definition at line 830 of file KDChartAbstractDiagram.cpp.

```
832 { return QModelIndex(); }
```

7.5.3.55 void AbstractDiagram::setAllowOverlappingDataValueTexts (bool *allow*) `[inherited]`

Set whether data value labels are allowed to overlap.

Parameters:

allow True means that overlapping labels are allowed.

Definition at line 440 of file KDChartAbstractDiagram.cpp.

References d.

```
445 {
```

7.5.3.56 void AbstractDiagram::setAntiAliasing (bool *enabled*) `[inherited]`

Set whether anti-aliasing is to be used while rendering this diagram.

Parameters:

enabled True means that AA is enabled.

Definition at line 451 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.5.3.57 void AbstractDiagram::setAttributesModel (AttributesModel ∗ *model*) `[virtual, inherited]`

Associate an AttributesModel with this diagram.

Note that the diagram does _not_ take ownership of the AttributesModel. This should thus only be used with AttributesModels that have been explicitely created by the user, and are owned by her. Setting an AttributesModel that is internal to another diagram is an error.

Correct:

```
AttributesModel *am = new AttributesModel( model, 0 );
diagram1->setAttributesModel( am );
diagram2->setAttributesModel( am );
```

Wrong:

```
diagram1->setAttributesModel( diagram2->attributesModel() );
```

Parameters:

model The AttributesModel to use for this diagram.

See also:

AttributesModel, usesExternalAttributesModel

Definition at line 261 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::AbstractDiagram::modelsChanged().

```
{
    if( amodel->sourceModel() != model() ) {
        qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
                 "Trying to set an attributesmodel which works on a different "
                 "model than the diagram.");
        return;
    }
    if( qobject_cast<PrivateAttributesModel*>(amodel) ) {
        qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
                 "Trying to set an attributesmodel that is private to another diagram.");
        return;
    }
    d->setAttributesModel(amodel);
    scheduleDelayedItemsLayout();
    d->databoundariesDirty = true;
    emit modelsChanged();
}
```

7.5.3.58 void AbstractDiagram::setAttributesModelRootIndex (const QModelIndex & *idx*) `[protected, inherited]`

Definition at line 301 of file KDChartAbstractDiagram.cpp.

References d.

7.5.3.59 void AbstractDiagram::setBrush (const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting all datasets in the model.

Parameters:

brush The brush to use.

Definition at line 786 of file KDChartAbstractDiagram.cpp.

```
792 {
```

7.5.3.60 void AbstractDiagram::setBrush (int *dataset*, const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting the given dataset.

Parameters:

dataset The dataset's column in the model.

pen The brush to use.

Definition at line 793 of file KDChartAbstractDiagram.cpp.

```
801 {
```

7.5.3.61 void AbstractDiagram::setBrush (const QModelIndex & *index*, const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting the datapoint at the given index.

Parameters:

index The datapoint's index in the model.

brush The brush to use.

Definition at line 778 of file KDChartAbstractDiagram.cpp.

```
785 {
```

7.5.3.62 void KDChart::AbstractCartesianDiagram::setCoordinatePlane (AbstractCoordinatePlane ∗ *plane*) `[virtual]`

Set the coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. The chart takes ownership.

Returns:

The coordinate plane associated with the diagram.

Reimplemented from KDChart::AbstractDiagram.

Definition at line 123 of file KDChartAbstractCartesianDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane(), and KDChart::AbstractDiagram::setCoordinatePlane().

```
124 {
125     if( coordinatePlane() ) disconnect( coordinatePlane() );
126     AbstractDiagram::setCoordinatePlane(plane);
127 
128     // show the axes, after all have been adjusted
129     // (because they might be dependend on each other)
130     /*
131     if( plane )
132         Q_FOREACH( CartesianAxis* axis, d->axesList )
133             axis->show();
134     else
135         Q_FOREACH( CartesianAxis* axis, d->axesList )
136             axis->hide();
137     */
138 }
```

7.5.3.63 void AbstractDiagram::setDataBoundariesDirty () const `[protected, inherited]`

Definition at line 240 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::BarDiagram::setThreeDBarAttributes(), KDChart::LineDiagram::setThreeDLineAttributes(), KDChart::LineDiagram::setType(), and KDChart::BarDiagram::setType().

```
241 {
242     d->databoundariesDirty = true;
243 }
```

7.5.3.64 void AbstractDiagram::setDatasetDimension (int *dimension*) `[inherited]`

Sets the dataset dimension of the diagram.

See also:

datasetDimension.

Parameters:

dimension

Definition at line 947 of file KDChartAbstractDiagram.cpp.

References d.

```
954 {
```

7.5.3.65 void AbstractDiagram::setDataValueAttributes (const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for all datapoints in the model.

Parameters:

a The attributes to set.

Definition at line 434 of file KDChartAbstractDiagram.cpp.

References d.

```
439 {
```

7.5.3.66 void AbstractDiagram::setDataValueAttributes (int *dataset*, const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for the given dataset.

Parameters:

dataset The dataset to set the attributes for.

a The attributes to set.

Definition at line 406 of file KDChartAbstractDiagram.cpp.

References d.

```
413 {
```

7.5.3.67 void AbstractDiagram::setDataValueAttributes (const QModelIndex & *index*, const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for the given index.

Parameters:

index The datapoint to set the attributes for.

a The attributes to set.

Definition at line 395 of file KDChartAbstractDiagram.cpp.

References d, KDChart::DataValueLabelAttributesRole, and KDChart::AbstractDiagram::propertiesChanged().

```
{
    d->attributesModel->setData(
        d->attributesModel->mapFromSource( index ),
        qVariantFromValue( a ),
        DataValueLabelAttributesRole );
    emit propertiesChanged();
}


```

7.5.3.68 void AbstractDiagram::setHidden (bool *hidden*) `[inherited]`

Hide (or unhide, resp.) all datapoints in the model.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

hidden The hidden status to set.

Definition at line 365 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.5.3.69 void AbstractDiagram::setHidden (int *column*, bool *hidden*) `[inherited]`

Hide (or unhide, resp.) a dataset.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

dataset The dataset to set the hidden status for.

hidden The hidden status to set.

Definition at line 356 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.5.3.70 void AbstractDiagram::setHidden (const QModelIndex & *index*, bool *hidden*) `[inherited]`

Hide (or unhide, resp.) a data cell.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

index The datapoint to set the hidden status for.

hidden The hidden status to set.

Definition at line 347 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::DataHiddenRole.

```
{
```

7.5.3.71 void AbstractDiagram::setModel (QAbstractItemModel ∗ *model*) `[virtual, inherited]`

Associate a model with the diagram.

Definition at line 245 of file KDChartAbstractDiagram.cpp.

References d, KDChart::AttributesModel::initFrom(), and KDChart::AbstractDiagram::modelsChanged().

```
{
  QAbstractItemView::setModel( newModel );
  AttributesModel* amodel = new PrivateAttributesModel( newModel, this );
  amodel->initFrom( d->attributesModel );
  d->setAttributesModel(amodel);
  scheduleDelayedItemsLayout();
  d->databoundariesDirty = true;
  emit modelsChanged();
}
```

7.5.3.72 void AbstractDiagram::setPen (const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting all datasets in the model.

Parameters:

pen The pen to use.

Definition at line 740 of file KDChartAbstractDiagram.cpp.

```
{
```

7.5.3.73 void AbstractDiagram::setPen (int *dataset*, const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting the given dataset.

Parameters:

dataset The dataset's row in the model.

pen The pen to use.

Definition at line 747 of file KDChartAbstractDiagram.cpp.

```
{
```

7.5.3.74 void AbstractDiagram::setPen (const QModelIndex & *index*, const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting the datapoint at the given index.

Parameters:

index The datapoint's index in the model.

pen The pen to use.

Definition at line 732 of file KDChartAbstractDiagram.cpp.

```
{
```

7.5.3.75 void AbstractDiagram::setPercentMode (bool *percent*) `[inherited]`

Definition at line 462 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::setType(), and KDChart::BarDiagram::setType().

```
{
```

7.5.3.76 void AbstractCartesianDiagram::setReferenceDiagram (AbstractCartesianDiagram ∗ *diagram*, const QPointF & *offset* = QPointF()) `[virtual]`

Definition at line 140 of file KDChartAbstractCartesianDiagram.cpp.

References d.

```
{
    d->referenceDiagram = diagram;
    d->referenceDiagramOffset = offset;
}
```

7.5.3.77 void AbstractDiagram::setRootIndex (const QModelIndex & *idx*) `[virtual, inherited]`

Set the root index in the model, where the diagram starts referencing data for display.

[reimplemented]

Definition at line 294 of file KDChartAbstractDiagram.cpp.

References d.

7.5.3.78 void AbstractDiagram::setSelection (const QRect & *rect*, QItemSelectionModel::SelectionFlags *command*) `[virtual, inherited]`

[reimplemented]

Definition at line 848 of file KDChartAbstractDiagram.cpp.

```
{ return QRegion(); }
```

7.5.3.79 void AbstractCartesianDiagram::takeAxis (CartesianAxis ∗ *axis*) `[virtual]`

Removes the axis from the diagram, without deleting it.

The diagram no longer owns the axis, so it is the caller's responsibility to delete the axis.

See also:
addAxis

Definition at line 98 of file KDChartAbstractCartesianDiagram.cpp.

References d, KDChart::AbstractAxis::deleteObserver(), layoutPlanes(), and KDChart::AbstractLayoutItem::setParentWidget().

Referenced by KDChart::CartesianAxis::∼CartesianAxis().

```
{
    const int idx = d->axesList.indexOf( axis );
    if( idx != -1 )
        d->axesList.takeAt( idx );
    axis->deleteObserver( this );
    axis->setParentWidget( 0 );
    layoutPlanes();
}
```

7.5.3.80 virtual double KDChart::AbstractCartesianDiagram::threeDItemDepth (int *column*) const `[protected, pure virtual]`

Implemented in KDChart::BarDiagram, and KDChart::LineDiagram.

7.5.3.81 virtual double KDChart::AbstractCartesianDiagram::threeDItemDepth (const QModelIndex & *index*) const `[protected, pure virtual]`

Implemented in KDChart::BarDiagram, and KDChart::LineDiagram.

7.5.3.82 void AbstractDiagram::update () const `[inherited]`

Definition at line 961 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractDiagram::doItemsLayout().

7.5.3.83 void KDChart::AbstractDiagram::useDefaultColors () `[inherited]`

Set the palette to be used, for painting datasets to the default palette.

See also:
KDChart::Palette. FIXME: fold into one usePalette (KDChart::Palette&) method

Definition at line 855 of file KDChartAbstractDiagram.cpp.

References d.

```
859 {
```

7.5.3.84 void KDChart::AbstractDiagram::useRainbowColors () `[inherited]`

Set the palette to be used, for painting datasets to the rainbow palette.

See also:
KDChart::Palette.

Definition at line 865 of file KDChartAbstractDiagram.cpp.

References d.

```
869 {
```

7.5.3.85 bool AbstractDiagram::usesExternalAttributesModel () const `[virtual, inherited]`

Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

See also:
setAttributesModel

Definition at line 280 of file KDChartAbstractDiagram.cpp.

References d.

```
281 {
282     return d->usesExternalAttributesModel();
283 }
```

7.5.3.86 void KDChart::AbstractDiagram::useSubduedColors () `[inherited]`

Set the palette to be used, for painting datasets to the subdued palette.

See also:

KDChart::Palette.

Definition at line 860 of file KDChartAbstractDiagram.cpp.

References d.

```
864 {
```

7.5.3.87 double AbstractDiagram::valueForCell (int *row*, int *column*) const `[protected, inherited]`

Helper method, retrieving the data value (DisplayRole) for a given row and column.

Parameters:

row The row to query.

column The column to query.

Returns:

The value of the display role at the given row and column as a double.

Definition at line 955 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::attributesModelRootIndex(), and d.

Referenced by KDChart::LineDiagram::paint().

```
960 {
```

7.5.3.88 int AbstractDiagram::verticalOffset () const `[virtual, inherited]`

[reimplemented]

Definition at line 842 of file KDChartAbstractDiagram.cpp.

```
844 { return true; }
```

7.5.3.89 QRect AbstractDiagram::visualRect (const QModelIndex & *index*) const `[virtual, inherited]`

[reimplemented]

Definition at line 825 of file KDChartAbstractDiagram.cpp.

```
829 {}
```

7.5.3.90 QRegion AbstractDiagram::visualRegionForSelection (const QItemSelection & *selection*) const `[virtual, inherited]`

[reimplemented]

Definition at line 851 of file KDChartAbstractDiagram.cpp.

7.5.4 Member Data Documentation

7.5.4.1 Q_SIGNALS KDChart::AbstractDiagram::__pad0__ `[protected, inherited]`

Definition at line 589 of file KDChartAbstractDiagram.h.

The documentation for this class was generated from the following files:

- KDChartAbstractCartesianDiagram.h
- KDChartAbstractCartesianDiagram.cpp

7.6 KDChart::AbstractCoordinatePlane Class Reference

`#include <KDChartAbstractCoordinatePlane.h>`

Inheritance diagram for KDChart::AbstractCoordinatePlane:Collaboration diagram for KDChart::AbstractCoordinatePlane:

Public Types

- enum AxesCalcMode {
  Linear,
  Logarithmic }

Public Member Functions

- virtual void addDiagram (AbstractDiagram ∗diagram)

  Adds a diagram to this coordinate plane.

- void alignToReferencePoint (const RelativePosition &position)
- BackgroundAttributes backgroundAttributes () const
- virtual int bottomOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart:AutoSpacerLayoutItem::sizeHint().

- bool compare (const AbstractAreaBase ∗other) const

  Returns true if both areas have the same settings.

- AbstractDiagram ∗ diagram ()
- ConstAbstractDiagramList diagrams () const
- AbstractDiagramList diagrams ()
- virtual Qt::Orientations expandingDirections () const

  pure virtual in QLayoutItem

- FrameAttributes frameAttributes () const
- virtual QRect geometry () const

  pure virtual in QLayoutItem

- void getFrameLeadings (int &left, int &top, int &right, int &bottom) const
- GridAttributes globalGridAttributes () const
- DataDimensionsList gridDimensionsList ()

  Returns the dimensions used for drawing the grid lines.

- virtual bool isEmpty () const

  pure virtual in QLayoutItem

- const bool isVisiblePoint (const QPointF &point) const

  Tests, if a point is visible on the coordinate plane.

- virtual void layoutDiagrams ()=0

  Distribute the available space among the diagrams and axes.

- void layoutPlanes ()

  Calling layoutPlanes() on the plane triggers the global KDChart::Chart::slotLayoutPlanes().

- virtual int leftOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- virtual QSize maximumSize () const

  pure virtual in QLayoutItem

- virtual QSize minimumSize () const

  pure virtual in QLayoutItem

- virtual QSize minimumSizeHint () const

  [reimplemented]

- void mousePressEvent (QMouseEvent ∗event)

  reimp

- void needLayoutPlanes ()

  Emitted when plane needs to trigger the Chart's layouting of the coord.

- void needRelayout ()

  Emitted when plane needs to trigger the Chart's layouting.

- void needUpdate ()

  Emitted when plane needs to update its drawings.

- virtual void paint (QPainter ∗)=0
- virtual void paintAll (QPainter &painter)

  Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

- virtual void paintBackground (QPainter &painter, const QRect &rectangle)
- virtual void paintCtx (PaintContext ∗context)

  Default impl: Paint the complete item using its layouted position and size.

- virtual void paintFrame (QPainter &painter, const QRect &rectangle)
- virtual void paintIntoRect (QPainter &painter, const QRect &rect)

  Draws the background and frame, then calls paint().

- const Chart ∗ parent () const
- Chart ∗ parent ()
- QLayout ∗ parentLayout ()
- void propertiesChanged ()

  Emitted upon change of a property of the Coordinate Plane or any of its components.

- AbstractCoordinatePlane ∗ referenceCoordinatePlane () const

  There are two ways, in which planes can be caused to interact, in where they are put layouting wise: The first is the reference plane.

- void relayout ()

  Calling relayout() on the plane triggers the global KDChart::Chart::slotRelayout().

- void removeFromParentLayout ()
- virtual void replaceDiagram (AbstractDiagram ∗diagram, AbstractDiagram ∗oldDiagram=0)

  Replaces the old diagram, or appends the diagram, it there is none yet.

- virtual int rightOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- void setBackgroundAttributes (const BackgroundAttributes &a)
- void setFrameAttributes (const FrameAttributes &a)
- virtual void setGeometry (const QRect &r)

  pure virtual in QLayoutItem

- void setGlobalGridAttributes (const GridAttributes &)

  Set the grid attributes to be used by this coordinate plane.

- void setGridNeedsRecalculate ()

  Used by the chart to clear the cached grid data.

- void setParent (Chart ∗parent)

  Called internally by KDChart::Chart.

- void setParentLayout (QLayout ∗lay)
- virtual void setParentWidget (QWidget ∗widget)

  Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

- void setReferenceCoordinatePlane (AbstractCoordinatePlane ∗plane)

  Set another coordinate plane to be used as the reference plane for this one.

- virtual void setZoomCenter (QPointF)

  Set the point (in value coordinates) to be used as the center point in zoom operations.

- virtual void setZoomFactorX (double)

  Sets the zoom factor in horizontal direction, that is applied to all coordinate transformations.

- virtual void setZoomFactorY (double)

  Sets the zoom factor in vertical direction, that is applied to all coordinate transformations.

- virtual QSize sizeHint () const

  pure virtual in QLayoutItem

- virtual void sizeHintChanged () const

  Report changed size hint: ask the parent widget to recalculate the layout.

- virtual QSizePolicy sizePolicy () const

  [reimplemented]

- virtual void takeDiagram (AbstractDiagram ∗diagram)

  Removes the diagram from the plane, without deleting it.

- virtual int topOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- virtual const QPointF translate (const QPointF &diagramPoint) const=0

  Translate the given point in value space coordinates to a position in pixel space.

- virtual QPointF zoomCenter () const
- virtual double zoomFactorX () const
- virtual double zoomFactorY () const
- virtual ∼AbstractCoordinatePlane ()

Static Public Member Functions

- void paintBackgroundAttributes (QPainter &painter, const QRect &rectangle, const KDChart::BackgroundAttributes &attributes)
- void paintFrameAttributes (QPainter &painter, const QRect &rectangle, const KDChart::FrameAttributes &attributes)

Public Attributes

- Q_SIGNALS __pad0__: void destroyedCoordinatePlane(AbstractCoordinatePlane∗)
- public Q_SLOTS: void update()

Protected Member Functions

- AbstractCoordinatePlane (Chart ∗parent=0)
- virtual QRect areaGeometry () const
- virtual DataDimensionsList getDataDimensionsList () const=0
- QRect innerRect () const
- virtual void positionHasChanged ()

Protected Attributes

- QWidget ∗ mParent
- QLayout ∗ mParentLayout

7.6.1 Member Enumeration Documentation

7.6.1.1 enum KDChart::AbstractCoordinatePlane::AxesCalcMode

Enumeration values:

Linear

Logarithmic

Definition at line 55 of file KDChartAbstractCoordinatePlane.h.

```
55 { Linear, Logarithmic };
```

7.6.2 Constructor & Destructor Documentation

7.6.2.1 AbstractCoordinatePlane::AbstractCoordinatePlane (Chart ∗ *parent* = 0) `[explicit, protected]`

Definition at line 51 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
52     : AbstractArea ( new Private() )
53 {
54     d->parent = parent;
55     d->init();
56 }
```

7.6.2.2 AbstractCoordinatePlane::∼AbstractCoordinatePlane () `[virtual]`

Definition at line 58 of file KDChartAbstractCoordinatePlane.cpp.

```
59 {
60     emit destroyedCoordinatePlane( this );
61 }
```

7.6.3 Member Function Documentation

7.6.3.1 void AbstractCoordinatePlane::addDiagram (AbstractDiagram ∗ *diagram*) `[virtual]`

Adds a diagram to this coordinate plane.

Parameters:

diagram The diagram to add.

See also:

replaceDiagram, takeDiagram

Reimplemented in KDChart::CartesianCoordinatePlane, and KDChart::PolarCoordinatePlane.

Definition at line 68 of file KDChartAbstractCoordinatePlane.cpp.

References d, layoutDiagrams(), layoutPlanes(), and KDChart::AbstractDiagram::setCoordinatePlane().

Referenced by KDChart::PolarCoordinatePlane::addDiagram(), KDChart::CartesianCoordinatePlane::addDiagram(), and replaceDiagram().

```
69 {
70     // diagrams are invisible and paint through their paint() method
71     diagram->hide();
72
73     d->diagrams.append( diagram );
74     diagram->setParent( d->parent );
```

```
    diagram->setCoordinatePlane( this );
    layoutDiagrams();
    layoutPlanes(); // there might be new axes, etc
    update();
}
```

7.6.3.2 void AbstractAreaBase::alignToReferencePoint (const RelativePosition & *position*) `[inherited]`

Definition at line 90 of file KDChartAbstractAreaBase.cpp.

```
{
    Q_UNUSED( position );
    // PENDING(kalle) FIXME
    qWarning( "Sorry, not implemented: void AbstractAreaBase::alignToReferencePoint( const RelativePosi
}
```

7.6.3.3 QRect AbstractArea::areaGeometry () const `[protected, virtual, inherited]`

Implements KDChart::AbstractAreaBase.

Definition at line 150 of file KDChartAbstractArea.cpp.

Referenced by KDChart::CartesianCoordinatePlane::drawingArea(), KDChart::PolarCoordinatePlane::layoutDiagrams(), KDChart::CartesianAxis::paint(), KDChart::AbstractArea::paintAll(), and KDChart::CartesianAxis::paintCtx().

```
{
    return geometry();
}
```

7.6.3.4 BackgroundAttributes AbstractAreaBase::backgroundAttributes () const `[inherited]`

Definition at line 112 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by updateCommonBrush().

```
{
    return d->backgroundAttributes;
}
```

7.6.3.5 int AbstractArea::bottomOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart:AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the bottom edge of the area.

Note:

The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 101 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
102 {
103     // Re-calculate the sizes,
104     // so we also get the amountOf..Overlap members set newly:
105     if( ! doNotRecalculate )
106         sizeHint();
107     return d->amountOfBottomOverlap;
108 }
```

7.6.3.6 bool AbstractAreaBase::compare (const AbstractAreaBase ∗ *other*) const `[inherited]`

Returns true if both areas have the same settings.

Definition at line 75 of file KDChartAbstractAreaBase.cpp.

```
76 {
77     if( other == this ) return true;
78     if( ! other ){
79         //qDebug() << "CartesianAxis::compare() cannot compare to Null pointer";
80         return false;
81     }
82     /*
83     qDebug() << "AbstractAreaBase:" << (frameAttributes() == other->frameAttributes())
84         << (backgroundAttributes() == other->backgroundAttributes()) << "\n";
85     */
86     return  (frameAttributes()      == other->frameAttributes()) &&
87             (backgroundAttributes() == other->backgroundAttributes());
88 }
```

7.6.3.7 AbstractDiagram ∗ AbstractCoordinatePlane::diagram ()

Returns:

The first diagram associated with this coordinate plane.

Definition at line 113 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::Widget::diagram(), KDChart::Chart::mousePressEvent(), and KDChart::PolarCoordinatePlane::setStartPosition().

```
114 {
115     if ( d->diagrams.isEmpty() )
116     {
117         return 0;
118     } else {
119         return d->diagrams.first();
120     }
121 }
```

7.6.3.8 ConstAbstractDiagramList AbstractCoordinatePlane::diagrams () const

Returns:

The list of diagrams associated with this coordinate plane.

Definition at line 128 of file KDChartAbstractCoordinatePlane.cpp.

References KDChart::ConstAbstractDiagramList, and d.

```
129 {
130     ConstAbstractDiagramList list;
131 #ifndef QT_NO_STL
132     qCopy( d->diagrams.begin(), d->diagrams.end(), std::back_inserter( list ) );
133 #else
134     Q_FOREACH( AbstractDiagram * a, d->diagrams )
135         list.push_back( a );
136 #endif
137     return list;
138 }
```

7.6.3.9 AbstractDiagramList AbstractCoordinatePlane::diagrams ()

Returns:

The list of diagrams associated with this coordinate plane.

Definition at line 123 of file KDChartAbstractCoordinatePlane.cpp.

References KDChart::AbstractDiagramList, and d.

Referenced by KDChart::CartesianCoordinatePlane::getDataDimensionsList(), KDChart::CartesianCoordinatePlane::getRawDataBoundingRectFromDiagrams(), KDChart::PolarCoordinatePlane::layoutDiagrams(), KDChart::CartesianCoordinatePlane::layoutDiagrams(), KDChart::Chart::mousePressEvent(), KDChart::PolarCoordinatePlane::paint(), and KDChart::CartesianCoordinatePlane::paint().

```
124 {
125     return d->diagrams;
126 }
```

7.6.3.10 Qt::Orientations KDChart::AbstractCoordinatePlane::expandingDirections () const `[virtual]`

pure virtual in QLayoutItem

Definition at line 208 of file KDChartAbstractCoordinatePlane.cpp.

```
209 {
210     return Qt::Vertical | Qt::Horizontal;
211 }
```

7.6.3.11 FrameAttributes AbstractAreaBase::frameAttributes () const `[inherited]`

Definition at line 102 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone(), and updateCommonBrush().

```
103 {
104     return d->frameAttributes;
105 }
```

7.6.3.12 QRect KDChart::AbstractCoordinatePlane::geometry () const `[virtual]`

pure virtual in QLayoutItem

Definition at line 242 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::Chart::mousePressEvent(), and KDChart::PolarCoordinatePlane::paint().

```
243 {
244     return d->geometry;
245 }
```

7.6.3.13 virtual DataDimensionsList KDChart::AbstractCoordinatePlane::getDataDimensionsList () const `[protected, pure virtual]`

Implemented in KDChart::CartesianCoordinatePlane, and KDChart::PolarCoordinatePlane.

7.6.3.14 void AbstractAreaBase::getFrameLeadings (int & *left*, int & *top*, int & *right*, int & *bottom*) const `[inherited]`

Definition at line 204 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::AbstractAreaBase::innerRect(), and KDChart::AbstractAreaWidget::paintAll().

```
205 {
206     if( d && d->frameAttributes.isVisible() ){
207         const int padding = qMax( d->frameAttributes.padding(), 0 );
208         left   = padding;
209         top    = padding;
210         right  = padding;
211         bottom = padding;
212     }else{
213         left   = 0;
214         top    = 0;
215         right  = 0;
216         bottom = 0;
217     }
218 }
```

7.6.3.15 GridAttributes KDChart::AbstractCoordinatePlane::globalGridAttributes () const

Returns:

The grid attributes used by this coordinate plane.

See also:

setGlobalGridAttributes
CartesianCoordinatePlane::gridAttributes

Definition at line 157 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::PolarCoordinatePlane::gridAttributes(), and KDChart::CartesianCoordinatePlane::gridAttributes().

```
{
    return d->gridAttributes;
}
```

7.6.3.16 KDChart::DataDimensionsList KDChart::AbstractCoordinatePlane::gridDimensionsList ()

Returns the dimensions used for drawing the grid lines.

Returned data is the result of (cached) grid calculations, so - if you need that information for your own tasks - make sure to call again this function after every data modification that has changed the data range, since grid calculation is based upon the data range, thus the grid start/end might have changed if the data was changed.

Note:

Returned list will contain different numbers of DataDimension, depending on the kind of coordinate plane used. For CartesianCoordinatePlane two DataDimension are returned: the first representing grid lines in X direction (matching the Abscissa axes) and the second indicating vertical grid lines (or Ordinate axes, resp.).

Returns:

The dimensions used for drawing the grid lines.

See also:

DataDimension

Definition at line 162 of file KDChartAbstractCoordinatePlane.cpp.

References d, and KDChart::DataDimensionsList.

Referenced by KDChart::CartesianCoordinatePlane::layoutDiagrams(), KDChart::CartesianAxis::maximumSize(), and KDChart::CartesianAxis::paintCtx().

```
{
    //KDChart::DataDimensionsList l( d->grid->updateData( this ) );
    //qDebug() << "AbstractCoordinatePlane::gridDimensionsList() Y-range:" << l.last().end - l.last().
    //qDebug() << "AbstractCoordinatePlane::gridDimensionsList() X-range:" << l.first().end - l.first(
    return d->grid->updateData( this );
}
```

7.6.3.17 QRect AbstractAreaBase::innerRect () const `[protected, inherited]`

Definition at line 220 of file KDChartAbstractAreaBase.cpp.

References KDChart::AbstractAreaBase::areaGeometry(), and KDChart::AbstractAreaBase::getFrameLeadings().

Referenced by KDChart::TextArea::paintAll(), and KDChart::AbstractArea::paintAll().

```
221 {
222     int left;
223     int top;
224     int right;
225     int bottom;
226     getFrameLeadings( left, top, right, bottom );
227     return
228         QRect( QPoint(0,0), areaGeometry().size() )
229             .adjusted( left, top, -right, -bottom );
230 }
```

7.6.3.18 bool KDChart::AbstractCoordinatePlane::isEmpty () const `[virtual]`

pure virtual in QLayoutItem

Definition at line 201 of file KDChartAbstractCoordinatePlane.cpp.

```
202 {
203     return false; // never empty!
204     // coordinate planes with no associated diagrams
205     // are showing a default grid of ()1..10, 1..10) stepWidth 1
206 }
```

7.6.3.19 const bool KDChart::AbstractCoordinatePlane::isVisiblePoint (const QPointF & *point*) const

Tests, if a point is visible on the coordinate plane.

Note:

Before calling this function the point must have been translated into coordinate plane space.

Definition at line 275 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
276 {
277     return d->isVisiblePoint( this, point );
278 }
```

7.6.3.20 virtual void KDChart::AbstractCoordinatePlane::layoutDiagrams () `[pure virtual]`

Distribute the available space among the diagrams and axes.

Implemented in KDChart::CartesianCoordinatePlane, and KDChart::PolarCoordinatePlane.

Referenced by addDiagram(), replaceDiagram(), and takeDiagram().

7.6.3.21 void KDChart::AbstractCoordinatePlane::layoutPlanes ()

Calling layoutPlanes() on the plane triggers the global KDChart::Chart::slotLayoutPlanes().

Definition at line 259 of file KDChartAbstractCoordinatePlane.cpp.

References needLayoutPlanes().

Referenced by addDiagram(), KDChart::CartesianAxis::layoutPlanes(), KDChart::AbstractCartesianDiagram::layoutPlanes(), and replaceDiagram().

```
260 {
261     //qDebug("KDChart::AbstractCoordinatePlane::relayout() called");
262     emit needLayoutPlanes();
263 }
```

7.6.3.22 int AbstractArea::leftOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the left edge of the area.

Note:

> The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 77 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
78 {
79     // Re-calculate the sizes,
80     // so we also get the amountOf..Overlap members set newly:
81     if( ! doNotRecalculate )
82         sizeHint();
83     return d->amountOfLeftOverlap;
84 }
```

7.6.3.23 QSize KDChart::AbstractCoordinatePlane::maximumSize () const `[virtual]`

pure virtual in QLayoutItem

Definition at line 213 of file KDChartAbstractCoordinatePlane.cpp.

Referenced by sizeHint().

```
214 {
215     // No maximum size set. Especially not parent()->size(), we are not layouting
216     // to the parent widget’s size when using Chart::paint()!
217     return QSize(QLAYOUTSIZE_MAX, QLAYOUTSIZE_MAX);
218 }
```

7.6.3.24 QSize KDChart::AbstractCoordinatePlane::minimumSize () const `[virtual]`

pure virtual in QLayoutItem

Definition at line 220 of file KDChartAbstractCoordinatePlane.cpp.

```
221 {
222     return QSize(60, 60); // this default can be overwritten by derived classes
223 }
```

7.6.3.25 QSize KDChart::AbstractCoordinatePlane::minimumSizeHint () const `[virtual]`

[reimplemented]

Definition at line 140 of file KDChartAbstractCoordinatePlane.cpp.

```
141 {
142     return QSize( 200, 200 );
143 }
```

7.6.3.26 void KDChart::AbstractCoordinatePlane::mousePressEvent (QMouseEvent ∗ *event*)

reimp

Definition at line 266 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::Chart::mousePressEvent().

```
267 {
268     KDAB_FOREACH( AbstractDiagram * a, d->diagrams )
269     {
270         a->mousePressEvent( event );
271     }
272 }
```

7.6.3.27 void KDChart::AbstractCoordinatePlane::needLayoutPlanes ()

Emitted when plane needs to trigger the Chart's layouting of the coord.

planes.

Referenced by layoutPlanes().

7.6.3.28 void KDChart::AbstractCoordinatePlane::needRelayout ()

Emitted when plane needs to trigger the Chart's layouting.

Referenced by relayout().

7.6.3.29 void KDChart::AbstractCoordinatePlane::needUpdate ()

Emitted when plane needs to update its drawings.

7.6.3.30 virtual void KDChart::AbstractLayoutItem::paint (QPainter ∗) `[pure virtual, inherited]`

Implemented in KDChart::CartesianAxis, KDChart::CartesianCoordinatePlane, KDChart::TextLayoutItem, KDChart::MarkerLayoutItem, KDChart::LineLayoutItem, KDChart::LineWithMarkerLayoutItem, KDChart::HorizontalLineLayoutItem, KDChart::VerticalLineLayoutItem, KDChart::AutoSpacerLayoutItem, and KDChart::PolarCoordinatePlane.

Referenced by KDChart::Legend::paint(), KDChart::AbstractLayoutItem::paintAll(), KDChart::AbstractArea::paintAll(), and KDChart::AbstractLayoutItem::paintCtx().

7.6.3.31 void AbstractArea::paintAll (QPainter & *painter*) `[virtual, inherited]`

Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

Reimplemented from KDChart::AbstractLayoutItem.

Definition at line 123 of file KDChartAbstractArea.cpp.

References KDChart::AbstractArea::areaGeometry(), d, KDChart::AbstractAreaBase::innerRect(), KDChart::AbstractLayoutItem::paint(), KDChart::AbstractAreaBase::paintBackground(), and KDChart::AbstractAreaBase::paintFrame().

Referenced by KDChart::AbstractArea::paintIntoRect().

```
{
    // Paint the background and frame
    const QRect overlappingArea( geometry().adjusted(
            -d->amountOfLeftOverlap,
            -d->amountOfTopOverlap,
            d->amountOfRightOverlap,
            d->amountOfBottomOverlap ) );
    paintBackground( painter, overlappingArea );
    paintFrame(      painter, overlappingArea );

    // temporarily adjust the widget size, to be sure all content gets calculated
    // to fit into the inner rectangle
    const QRect oldGeometry( areaGeometry()  );
    QRect inner( innerRect() );
    inner.moveTo(
        oldGeometry.left() + inner.left(),
        oldGeometry.top()  + inner.top() );
    const bool needAdjustGeometry = oldGeometry != inner;
    if( needAdjustGeometry )
        setGeometry( inner );
    paint( &painter );
    if( needAdjustGeometry )
        setGeometry( oldGeometry );
    //qDebug() << "AbstractAreaWidget::paintAll() done.";
}
```

7.6.3.32 void AbstractAreaBase::paintBackground (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 188 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintBackgroundAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintBackground()",
                "Private class was not initialized!" );
    paintBackgroundAttributes( painter, rect, d->backgroundAttributes );
}
```

7.6.3.33 void AbstractAreaBase::paintBackgroundAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::BackgroundAttributes & *attributes*) `[static, inherited]`

Definition at line 119 of file KDChartAbstractAreaBase.cpp.

References KDChart::BackgroundAttributes::brush(), KDChart::BackgroundAttributes::isVisible(), KDChart::BackgroundAttributes::pixmap(), and KDChart::BackgroundAttributes::pixmapMode().

Referenced by KDChart::AbstractAreaBase::paintBackground().

```
{
    if( !attributes.isVisible() ) return;

    /* first draw the brush (may contain a pixmap)*/
    if( Qt::NoBrush != attributes.brush().style() ) {
        KDChart::PainterSaver painterSaver( &painter );
        painter.setPen( Qt::NoPen );
        const QPointF newTopLeft( painter.deviceMatrix().map( rect.topLeft() ) );
        painter.setBrushOrigin( newTopLeft );
        painter.setBrush( attributes.brush() );
        painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
    }
    /* next draw the backPixmap over the brush */
    if( !attributes.pixmap().isNull() &&
        attributes.pixmapMode() != BackgroundAttributes::BackgroundPixmapModeNone ) {
        QPointF ol = rect.topLeft();
        if( BackgroundAttributes::BackgroundPixmapModeCentered == attributes.pixmapMode() )
        {
            ol.setX( rect.center().x() - attributes.pixmap().width() / 2 );
            ol.setY( rect.center().y() - attributes.pixmap().height()/ 2 );
            painter.drawPixmap( ol, attributes.pixmap() );
        } else {
            QMatrix m;
            double zW = (double)rect.width()  / (double)attributes.pixmap().width();
            double zH = (double)rect.height() / (double)attributes.pixmap().height();
            switch( attributes.pixmapMode() ) {
            case BackgroundAttributes::BackgroundPixmapModeScaled:
            {
                double z;
                z = qMin( zW, zH );
                m.scale( z, z );
            }
            break;
            case BackgroundAttributes::BackgroundPixmapModeStretched:
                m.scale( zW, zH );
                break;
            default:
                ; // Cannot happen, previously checked
            }
            QPixmap pm = attributes.pixmap().transformed( m );
            ol.setX( rect.center().x() - pm.width() / 2 );
            ol.setY( rect.center().y() - pm.height()/ 2 );
            painter.drawPixmap( ol, pm );
        }
    }
}
```

7.6.3.34 void KDChart::AbstractLayoutItem::paintCtx (PaintContext ∗ *context*) `[virtual, inherited]`

Default impl: Paint the complete item using its layouted position and size.

Reimplemented in KDChart::CartesianAxis.

Definition at line 77 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint(), and KDChart::PaintContext::painter().

```
{
    if( context )
        paint( context->painter() );
}
```

7.6.3.35 void AbstractAreaBase::paintFrame (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 196 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintFrameAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintFrame()",
                "Private class was not initialized!" );
    paintFrameAttributes( painter, rect, d->frameAttributes );
}
```

7.6.3.36 void AbstractAreaBase::paintFrameAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::FrameAttributes & *attributes*) `[static, inherited]`

Definition at line 169 of file KDChartAbstractAreaBase.cpp.

References KDChart::FrameAttributes::isVisible(), and KDChart::FrameAttributes::pen().

Referenced by KDChart::AbstractAreaBase::paintFrame().

```
{

    if( !attributes.isVisible() ) return;

    // Note: We set the brush to NoBrush explicitely here.
    //       Otherwise we might get a filled rectangle, so any
    //       previously drawn background would be overwritten by that area.

    const QPen   oldPen(   painter.pen() );
    const QBrush oldBrush( painter.brush() );
    painter.setPen(   attributes.pen() );
    painter.setBrush( Qt::NoBrush );
    painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
    painter.setBrush( oldBrush );
    painter.setPen(   oldPen );
}
```

7.6.3.37 void AbstractArea::paintIntoRect (QPainter & *painter*, const QRect & *rect*) `[virtual, inherited]`

Draws the background and frame, then calls paint().

In most cases there is no need to overwrite this method in a derived class, but you would overwrite AbstractLayoutItem::paint() instead.

Definition at line 111 of file KDChartAbstractArea.cpp.

References KDChart::AbstractArea::paintAll().

```
112 {
113     const QRect oldGeometry( geometry() );
114     if( oldGeometry != rect )
115         setGeometry( rect );
116     painter.translate( rect.left(), rect.top() );
117     paintAll( painter );
118     painter.translate( -rect.left(), -rect.top() );
119     if( oldGeometry != rect )
120         setGeometry( oldGeometry );
121 }
```

7.6.3.38 const KDChart::Chart ∗ KDChart::AbstractCoordinatePlane::parent () const

Definition at line 190 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
191 {
192     return d->parent;
193 }
```

7.6.3.39 KDChart::Chart ∗ KDChart::AbstractCoordinatePlane::parent ()

Definition at line 195 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::CartesianAxis::maximumSize(), and KDChart::CartesianAxis::paintCtx().

```
196 {
197     return d->parent;
198 }
```

7.6.3.40 QLayout∗ KDChart::AbstractLayoutItem::parentLayout () `[inherited]`

Definition at line 74 of file KDChartLayoutItems.h.

```
75          {
76              return mParentLayout;
77          }
```

7.6.3.41 void AbstractArea::positionHasChanged () `[protected, virtual, inherited]`

Reimplemented from KDChart::AbstractAreaBase.

Definition at line 155 of file KDChartAbstractArea.cpp.

```
156 {
157     emit positionChanged( this );
158 }
```

7.6.3.42 void KDChart::AbstractCoordinatePlane::propertiesChanged ()

Emitted upon change of a property of the Coordinate Plane or any of its components.

Referenced by KDChart::CartesianCoordinatePlane::addDiagram(), KDChart::CartesianCoordinatePlane::adjustHorizontalRangeToData(), KDChart::CartesianCoordinatePlane::adjustVerticalRangeToData(), KDChart::CartesianCoordinatePlane::setAutoAdjustGridToZoom(), KDChart::CartesianCoordinatePlane::setAutoAdjustHorizontalRangeToData(), KDChart::CartesianCoordinatePlane::setAutoAdjustVerticalRangeToData(), KDChart::CartesianCoordinatePlane::setAxesCalcModes(), KDChart::CartesianCoordinatePlane::setAxesCalcModeX(), KDChart::CartesianCoordinatePlane::setAxesCalcModeY(), KDChart::PolarCoordinatePlane::setGridAttributes(), KDChart::CartesianCoordinatePlane::setGridAttributes(), KDChart::CartesianCoordinatePlane::setHorizontalRange(), KDChart::CartesianCoordinatePlane::setIsometricScaling(), KDChart::CartesianCoordinatePlane::setVerticalRange(), KDChart::CartesianCoordinatePlane::setZoomCenter(), KDChart::CartesianCoordinatePlane::setZoomFactorX(), and KDChart::CartesianCoordinatePlane::setZoomFactorY().

7.6.3.43 AbstractCoordinatePlane ∗ KDChart::AbstractCoordinatePlane::referenceCoordinatePlane () const

There are two ways, in which planes can be caused to interact, in where they are put layouting wise: The first is the reference plane.

If such a reference plane is set, on a plane, it will use the same cell in the layout as that one. In addition to this, planes can share an axis. In that case they will be layed out in relation to each other as suggested by the position of the axis. If, for example Plane1 and Plane2 share an axis at position Left, that will result in the layout: Axis Plane1 Plane 2, vertically. If Plane1 also happens to be Plane2's reference plane, both planes are drawn over each other. The reference plane concept allows two planes to share the same space even if neither has any axis, and in case there are shared axis, it is used to decided, whether the planes should be painted on top of each other or layed out vertically or horizontally next to each other.

Returns:

The reference coordinate plane associated with this one.

Definition at line 180 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
{
    return d->referenceCoordinatePlane;
}
```

7.6.3.44 void KDChart::AbstractCoordinatePlane::relayout ()

Calling relayout() on the plane triggers the global KDChart::Chart::slotRelayout().

Definition at line 253 of file KDChartAbstractCoordinatePlane.cpp.

References needRelayout().

```
{
    //qDebug("KDChart::AbstractCoordinatePlane::relayout() called");
    emit needRelayout();
}
```

7.6.3.45 void KDChart::AbstractLayoutItem::removeFromParentLayout () `[inherited]`

Definition at line 78 of file KDChartLayoutItems.h.

Referenced by KDChart::Chart::takeCoordinatePlane().

```
        {
            if( mParentLayout ){
                if( widget() )
                    mParentLayout->removeWidget( widget() );
                else
                    mParentLayout->removeItem( this );
            }
        }
```

7.6.3.46 void AbstractCoordinatePlane::replaceDiagram (AbstractDiagram ∗ *diagram*, AbstractDiagram ∗ *oldDiagram* = 0) `[virtual]`

Replaces the old diagram, or appends the diagram, it there is none yet.

Parameters:

diagram The diagram to be used instead of the old diagram. This parameter must not be zero, or the method will do nothing.

oldDiagram The diagram to be removed by the new diagram. This diagram will be deleted automatically. If the parameter is omitted, the very first diagram will be replaced. In case, there was no diagram yet, the new diagram will just be added.

Note:

If you want to re-use the old diagram, call takeDiagram and addDiagram, instead of using replaceDiagram.

See also:

addDiagram, takeDiagram

Definition at line 82 of file KDChartAbstractCoordinatePlane.cpp.

References addDiagram(), d, layoutDiagrams(), layoutPlanes(), and takeDiagram().

```
{
    if( diagram && oldDiagram_ != diagram ){
        AbstractDiagram* oldDiagram = oldDiagram_;
        if( d->diagrams.count() ){
            if( ! oldDiagram )
                oldDiagram = d->diagrams.first();
            takeDiagram( oldDiagram );
        }
        delete oldDiagram;
        addDiagram( diagram );
        layoutDiagrams();
        layoutPlanes(); // there might be new axes, etc
        update();
    }
}
```

7.6.3.47 int AbstractArea::rightOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the right edge of the area.

Note:

The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 85 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
    // Re-calculate the sizes,
    // so we also get the amountOf..Overlap members set newly:
    if( ! doNotRecalculate )
        sizeHint();
    return d->amountOfRightOverlap;
}
```

7.6.3.48 void AbstractAreaBase::setBackgroundAttributes (const BackgroundAttributes & *a*) `[inherited]`

Definition at line 107 of file KDChartAbstractAreaBase.cpp.

References d.

```
{
    d->backgroundAttributes = a;
}
```

7.6.3.49 void AbstractAreaBase::setFrameAttributes (const FrameAttributes & *a*) `[inherited]`

Definition at line 97 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone().

```
{
    d->frameAttributes = a;
}
```

7.6.3.50 void KDChart::AbstractCoordinatePlane::setGeometry (const QRect & *r*) `[virtual]`

pure virtual in QLayoutItem

Note:

Do not call this function directly, unless you know exactly what you are doing. Geometry management is done by KD Chart's internal layouting measures.

Definition at line 232 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
{
//    qDebug() << "KDChart::AbstractCoordinatePlane::setGeometry(" << r << ") called";
    if( d->geometry != r ){
        d->geometry = r;
        // Note: We do *not* call update() here
        //       because it would invoke KDChart::update() recursively.
    }
}
```

7.6.3.51 void KDChart::AbstractCoordinatePlane::setGlobalGridAttributes (const GridAttributes &)

Set the grid attributes to be used by this coordinate plane.

To disable grid painting, for example, your code should like this:

```
GridAttributes ga = plane->globalGridAttributes();
ga.setGlobalGridVisible( false );
plane->setGlobalGridAttributes( ga );
```

See also:

globalGridAttributes
CartesianCoordinatePlane::setGridAttributes

Definition at line 151 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
{
    d->gridAttributes = a;
    update();
}
```

7.6.3.52 void KDChart::AbstractCoordinatePlane::setGridNeedsRecalculate ()

Used by the chart to clear the cached grid data.

Definition at line 170 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::Chart::resizeEvent().

```
{
    d->grid->setNeedRecalculate();
}
```

7.6.3.53 void KDChart::AbstractCoordinatePlane::setParent (Chart ∗ *parent*)

Called internally by KDChart::Chart.

Definition at line 185 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::Chart::addCoordinatePlane(), and KDChart::Chart::takeCoordinatePlane().

```
{
    d->parent = parent;
}
```

7.6.3.54 void KDChart::AbstractLayoutItem::setParentLayout (QLayout ∗ *lay*) `[inherited]`

Definition at line 70 of file KDChartLayoutItems.h.

```
        {
            mParentLayout = lay;
        }
```

7.6.3.55 void KDChart::AbstractLayoutItem::setParentWidget (QWidget ∗ *widget*) `[virtual, inherited]`

Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

Thus, you need to call setParentWidget on every item, that has a non-fixed size.

Definition at line 64 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::mParent.

Referenced by KDChart::Legend::buildLegend(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
{
    mParent = widget;
}
```

7.6.3.56 void KDChart::AbstractCoordinatePlane::setReferenceCoordinatePlane (AbstractCoordinatePlane ∗ *plane*)

Set another coordinate plane to be used as the reference plane for this one.

Parameters:

plane The coordinate plane to be used the reference plane for this one.

See also:

referenceCoordinatePlane

Definition at line 175 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
{
    d->referenceCoordinatePlane = plane;
}
```

7.6.3.57 virtual void KDChart::AbstractCoordinatePlane::setZoomCenter (QPointF) `[virtual]`

Set the point (in value coordinates) to be used as the center point in zoom operations.

Parameters:

center The point to use.

Reimplemented in KDChart::CartesianCoordinatePlane, and KDChart::PolarCoordinatePlane.

Definition at line 172 of file KDChartAbstractCoordinatePlane.h.

```
172 :
         * \code
```

7.6.3.58 virtual void KDChart::AbstractCoordinatePlane::setZoomFactorX (double) `[virtual]`

Sets the zoom factor in horizontal direction, that is applied to all coordinate transformations.

Reimplemented in KDChart::CartesianCoordinatePlane, and KDChart::PolarCoordinatePlane.

Definition at line 153 of file KDChartAbstractCoordinatePlane.h.

```
155 {}
```

7.6.3.59 virtual void KDChart::AbstractCoordinatePlane::setZoomFactorY (double) `[virtual]`

Sets the zoom factor in vertical direction, that is applied to all coordinate transformations.

Reimplemented in KDChart::CartesianCoordinatePlane, and KDChart::PolarCoordinatePlane.

Definition at line 159 of file KDChartAbstractCoordinatePlane.h.

```
161 { return QPointF(0.0, 0.0); }
```

7.6.3.60 QSize KDChart::AbstractCoordinatePlane::sizeHint () const `[virtual]`

pure virtual in QLayoutItem

Definition at line 225 of file KDChartAbstractCoordinatePlane.cpp.

References maximumSize().

```
226 {
227     // we return our maxiumu (which is the full size of the Chart)
228     // even if we know the plane will be smaller
229     return maximumSize();
230 }
```

7.6.3.61 void KDChart::AbstractLayoutItem::sizeHintChanged () const `[virtual, inherited]`

Report changed size hint: ask the parent widget to recalculate the layout.

Definition at line 86 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextLayoutItem::sizeHint().

```
87 {
88     // This is exactly like what QWidget::updateGeometry does.
89 //  qDebug("KDChart::AbstractLayoutItem::sizeHintChanged() called");
90     if( mParent ) {
91         if ( mParent->layout() )
92             mParent->layout()->invalidate();
93         else
94             QApplication::postEvent( mParent, new QEvent( QEvent::LayoutRequest ) );
95     }
96 }
```

7.6.3.62 QSizePolicy KDChart::AbstractCoordinatePlane::sizePolicy () const `[virtual]`

[reimplemented]

Definition at line 146 of file KDChartAbstractCoordinatePlane.cpp.

```
147 {
148     return QSizePolicy( QSizePolicy::MinimumExpanding, QSizePolicy::MinimumExpanding );
149 }
```

7.6.3.63 void AbstractCoordinatePlane::takeDiagram (AbstractDiagram ∗ *diagram*) `[virtual]`

Removes the diagram from the plane, without deleting it.

The plane no longer owns the diagram, so it is the caller’s responsibility to delete the diagram.

See also:

addDiagram, replaceDiagram

Definition at line 100 of file KDChartAbstractCoordinatePlane.cpp.

References d, layoutDiagrams(), and KDChart::AbstractDiagram::setCoordinatePlane().

Referenced by replaceDiagram().

```
101 {
102     const int idx = d->diagrams.indexOf( diagram );
103     if( idx != -1 ){
104         d->diagrams.removeAt( idx );
105         diagram->setParent( 0 );
106         diagram->setCoordinatePlane( 0 );
107         layoutDiagrams();
108         update();
109     }
110 }
```

7.6.3.64 int AbstractArea::topOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the top edge of the area.

Note:

The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 93 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
    // Re-calculate the sizes,
    // so we also get the amountOf..Overlap members set newly:
    if( ! doNotRecalculate )
        sizeHint();
    return d->amountOfTopOverlap;
}
```

7.6.3.65 virtual const QPointF KDChart::AbstractCoordinatePlane::translate (const QPointF & *diagramPoint*) const `[pure virtual]`

Translate the given point in value space coordinates to a position in pixel space.

Parameters:

diagramPoint The point in value coordinates.

Returns:

The translated point.

Implemented in KDChart::CartesianCoordinatePlane, and KDChart::PolarCoordinatePlane.

Referenced by KDChart::PolarDiagram::paint(), and KDChart::LineDiagram::paint().

7.6.3.66 virtual QPointF KDChart::AbstractCoordinatePlane::zoomCenter () const `[virtual]`

Returns:

The center point (in value coordinates) of the coordinate plane, that is used for zoom operations.

Reimplemented in KDChart::CartesianCoordinatePlane, and KDChart::PolarCoordinatePlane.

Definition at line 165 of file KDChartAbstractCoordinatePlane.h.

```
{}
```

7.6.3.67 virtual double KDChart::AbstractCoordinatePlane::zoomFactorX () const `[virtual]`

Returns:

The zoom factor in horizontal direction, that is applied to all coordinate transformations.

Reimplemented in KDChart::CartesianCoordinatePlane, and KDChart::PolarCoordinatePlane.

Definition at line 141 of file KDChartAbstractCoordinatePlane.h.

```
143 { return 1.0; }
```

7.6.3.68 virtual double KDChart::AbstractCoordinatePlane::zoomFactorY () const `[virtual]`

Returns:

The zoom factor in vertical direction, that is applied to all coordinate transformations.

Reimplemented in KDChart::CartesianCoordinatePlane, and KDChart::PolarCoordinatePlane.

Definition at line 147 of file KDChartAbstractCoordinatePlane.h.

```
149 {}
```

7.6.4 Member Data Documentation

7.6.4.1 Q_SIGNALS KDChart::AbstractCoordinatePlane::__pad0__

Reimplemented from KDChart::AbstractArea.

Definition at line 297 of file KDChartAbstractCoordinatePlane.h.

7.6.4.2 QWidget∗ KDChart::AbstractLayoutItem::mParent `[protected, inherited]`

Definition at line 88 of file KDChartLayoutItems.h.

Referenced by KDChart::AbstractLayoutItem::setParentWidget().

7.6.4.3 QLayout∗ KDChart::AbstractLayoutItem::mParentLayout `[protected, inherited]`

Definition at line 89 of file KDChartLayoutItems.h.

7.6.4.4 public KDChart::AbstractCoordinatePlane::Q_SLOTS

Reimplemented in KDChart::CartesianCoordinatePlane, KDChart::CartesianCoordinatePlane, and KDChart::PolarCoordinatePlane.

Definition at line 281 of file KDChartAbstractCoordinatePlane.h.

The documentation for this class was generated from the following files:

- KDChartAbstractCoordinatePlane.h
- KDChartAbstractCoordinatePlane.cpp

7.7 KDChart::AbstractDiagram Class Reference

`#include <KDChartAbstractDiagram.h>`

Inheritance diagram for KDChart::AbstractDiagram:Collaboration diagram for KDChart::AbstractDiagram:

7.7.1 Detailed Description

AbstractDiagram defines the interface for diagram classes.

AbstractDiagram is the base class for diagram classes ("chart types").

It defines the interface, that needs to be implemented for the diagram, to function within the KDChart framework. It extends Interview's QAbstractItemView.

Definition at line 53 of file KDChartAbstractDiagram.h.

Public Member Functions

- bool allowOverlappingDataValueTexts () const
- bool antiAliasing () const
- virtual AttributesModel $*$ attributesModel () const

  Returns the AttributesModel, that is used by this diagram.

- QBrush brush (const QModelIndex &index) const

  Retrieve the brush to be used, for painting the datapoint at the given index in the model.

- QBrush brush (int dataset) const

  Retrieve the brush to be used for the given dataset.

- QBrush brush () const

  Retrieve the brush to be used for painting datapoints globally.

- bool compare (const AbstractDiagram $*$other) const

  Returns true if both diagrams have the same settings.

- AbstractCoordinatePlane $*$ coordinatePlane () const

  The coordinate plane associated with the diagram.

- const QPair< QPointF, QPointF > dataBoundaries () const

  Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

- virtual void dataChanged (const QModelIndex &topLeft, const QModelIndex &bottomRight)

  [reimplemented]

- QList< QBrush > datasetBrushes () const

  The set of dataset brushes currently used, for use in legends, etc.

- int datasetDimension () const

The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

- QStringList datasetLabels () const

  The set of dataset labels currently displayed, for use in legends, etc.

- QList< MarkerAttributes > datasetMarkers () const

  The set of dataset markers currently used, for use in legends, etc.

- QList< QPen > datasetPens () const

  The set of dataset pens currently used, for use in legends, etc.

- DataValueAttributes dataValueAttributes (const QModelIndex &index) const

  Retrieve the DataValueAttributes for the given index.

- DataValueAttributes dataValueAttributes (int column) const

  Retrieve the DataValueAttributes for the given dataset.

- DataValueAttributes dataValueAttributes () const

  Retrieve the DataValueAttributes speficied globally.

- virtual void doItemsLayout ()

  [reimplemented]

- virtual int horizontalOffset () const

  [reimplemented]

- virtual QModelIndex indexAt (const QPoint &point) const

  [reimplemented]

- bool isHidden (const QModelIndex &index) const

  Retrieve the hidden status for the given index.

- bool isHidden (int column) const

  Retrieve the hidden status for the given dataset.

- bool isHidden () const

  Retrieve the hidden status speficied globally.

- virtual bool isIndexHidden (const QModelIndex &index) const

  [reimplemented]

- QStringList itemRowLabels () const

  The set of item row labels currently displayed, for use in Abscissa axes, etc.

- virtual QModelIndex moveCursor (CursorAction cursorAction, Qt::KeyboardModifiers modifiers)

  [reimplemented]

- virtual void paint (PaintContext ∗paintContext)=0

  Draw the diagram contents to the rectangle and painter, that are passed in as part of the paint context.

- void paintDataValueText (QPainter ∗painter, const QModelIndex &index, const QPointF &pos, double value)
- void paintMarker (QPainter ∗painter, const QModelIndex &index, const QPointF &pos)
- virtual void paintMarker (QPainter ∗painter, const MarkerAttributes &markerAttributes, const QBrush &brush, const QPen &, const QPointF &point, const QSizeF &size)
- QPen pen (const QModelIndex &index) const

  Retrieve the pen to be used, for painting the datapoint at the given index in the model.

- QPen pen (int dataset) const

  Retrieve the pen to be used for the given dataset.

- QPen pen () const

  Retrieve the pen to be used for painting datapoints globally.

- bool percentMode () const
- virtual void resize (const QSizeF &area)=0

  Called by the widget's sizeEvent.

- virtual void scrollTo (const QModelIndex &index, ScrollHint hint=EnsureVisible)

  [reimplemented]

- void setAllowOverlappingDataValueTexts (bool allow)

  Set whether data value labels are allowed to overlap.

- void setAntiAliasing (bool enabled)

  Set whether anti-aliasing is to be used while rendering this diagram.

- virtual void setAttributesModel (AttributesModel ∗model)

  Associate an AttributesModel with this diagram.

- void setBrush (const QBrush &brush)

  Set the brush to be used, for painting all datasets in the model.

- void setBrush (int dataset, const QBrush &brush)

  Set the brush to be used, for painting the given dataset.

- void setBrush (const QModelIndex &index, const QBrush &brush)

  Set the brush to be used, for painting the datapoint at the given index.

- virtual void setCoordinatePlane (AbstractCoordinatePlane ∗plane)

  Set the coordinate plane associated with the diagram.

- void setDatasetDimension (int dimension)

  Sets the dataset dimension of the diagram.

- void setDataValueAttributes (const DataValueAttributes &a)

  Set the DataValueAttributes for all datapoints in the model.

- void setDataValueAttributes (int dataset, const DataValueAttributes &a)

Set the DataValueAttributes for the given dataset.

- void setDataValueAttributes (const QModelIndex &index, const DataValueAttributes &a)

  Set the DataValueAttributes for the given index.

- void setHidden (bool hidden)

  Hide (or unhide, resp.) all datapoints in the model.

- void setHidden (int column, bool hidden)

  Hide (or unhide, resp.) a dataset.

- void setHidden (const QModelIndex &index, bool hidden)

  Hide (or unhide, resp.) a data cell.

- virtual void setModel (QAbstractItemModel $*$model)

  Associate a model with the diagram.

- void setPen (const QPen &pen)

  Set the pen to be used, for painting all datasets in the model.

- void setPen (int dataset, const QPen &pen)

  Set the pen to be used, for painting the given dataset.

- void setPen (const QModelIndex &index, const QPen &pen)

  Set the pen to be used, for painting the datapoint at the given index.

- void setPercentMode (bool percent)
- virtual void setRootIndex (const QModelIndex &idx)

  Set the root index in the model, where the diagram starts referencing data for display.

- virtual void setSelection (const QRect &rect, QItemSelectionModel::SelectionFlags command)

  [reimplemented]

- void update () const
- void useDefaultColors ()

  Set the palette to be used, for painting datasets to the default palette.

- void useRainbowColors ()

  Set the palette to be used, for painting datasets to the rainbow palette.

- virtual bool usesExternalAttributesModel () const

  Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

- void useSubduedColors ()

  Set the palette to be used, for painting datasets to the subdued palette.

- virtual int verticalOffset () const

  [reimplemented]

- virtual QRect visualRect (const QModelIndex &index) const

  [reimplemented]

- virtual QRegion visualRegionForSelection (const QItemSelection &selection) const

  [reimplemented]

- virtual ∼AbstractDiagram ()

Protected Member Functions

- AbstractDiagram (QWidget ∗parent=0, AbstractCoordinatePlane ∗plane=0)
- AbstractDiagram (Private ∗p, QWidget ∗parent, AbstractCoordinatePlane ∗plane)
- QModelIndex attributesModelRootIndex () const
- virtual const QPair< QPointF, QPointF > calculateDataBoundaries () const=0
- virtual bool checkInvariants (bool justReturnTheStatus=false) const
- QModelIndex columnToIndex (int column) const
- void dataHidden ()

  This signal is emitted, when the hidden status of at least one data cell was (un)set.

- void modelsChanged ()

  This signal is emitted, when either the model or the AttributesModel is replaced.

- virtual void paintDataValueTexts (QPainter ∗painter)
- virtual void paintMarkers (QPainter ∗painter)
- void propertiesChanged ()

  Emitted upon change of a property of the Diagram.

- void setAttributesModelRootIndex (const QModelIndex &)
- void setDataBoundariesDirty () const
- double valueForCell (int row, int column) const

  Helper method, retrieving the data value (DisplayRole) for a given row and column.

Protected Attributes

- Q_SIGNALS __pad0__: void layoutChanged(AbstractDiagram∗)

7.7.2 Constructor & Destructor Documentation

7.7.2.1 KDChart::AbstractDiagram::AbstractDiagram (Private ∗ *p*, QWidget ∗ *parent*, AbstractCoordinatePlane ∗ *plane*) `[explicit, protected]`

7.7.2.2 AbstractDiagram::AbstractDiagram (QWidget ∗ *parent* = 0, AbstractCoordinatePlane ∗ *plane* = 0) `[explicit, protected]`

Definition at line 119 of file KDChartAbstractDiagram.cpp.

```
120     : QAbstractItemView ( parent ), _d( new Private() )
121 {
122     _d->init( plane );
123 }
```

7.7.2.3 AbstractDiagram::∼AbstractDiagram () `[virtual]`

Definition at line 125 of file KDChartAbstractDiagram.cpp.

```
126 {
127     delete _d;
128 }
```

7.7.3 Member Function Documentation

7.7.3.1 bool AbstractDiagram::allowOverlappingDataValueTexts () const

Returns:
Whether data value labels are allowed to overlap.

Definition at line 446 of file KDChartAbstractDiagram.cpp.

References d.

```
450 {
```

7.7.3.2 bool AbstractDiagram::antiAliasing () const

Returns:
Whether anti-aliasing is to be used for rendering this diagram.

Definition at line 457 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::paint().

```
461 {
```

7.7.3.3 AttributesModel $*$ AbstractDiagram::attributesModel () const `[virtual]`

Returns the AttributesModel, that is used by this diagram.

By default each diagram owns its own AttributesModel, which should never be deleted. Only if a user-supplied AttributesModel has been set does the pointer returned here not belong to the diagram.

Returns:
The AttributesModel associated with the diagram.

See also:
setAttributesModel

Definition at line 286 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::RingDiagram::paint(), KDChart::PolarDiagram::paint(), and KDChart::BarDiagram::setBarAttributes().

```
287 {
288     return d->attributesModel;
289 }
```

7.7.3.4 QModelIndex AbstractDiagram::attributesModelRootIndex () const `[protected]`

returns a QModelIndex pointing into the AttributesModel that corresponds to the root index of the diagram.

Definition at line 310 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::calculateDataBoundaries(), KDChart::BarDiagram::calculateDataBoundaries(), KDChart::LineDiagram::numberOfAbscissaSegments(), KDChart::BarDiagram::numberOfAbscissaSegments(), KDChart::LineDiagram::numberOfOrdinateSegments(), KDChart::BarDiagram::numberOfOrdinateSegments(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and valueForCell().

```
316 {
```

7.7.3.5 QBrush AbstractDiagram::brush (const QModelIndex & *index*) const

Retrieve the brush to be used, for painting the datapoint at the given index in the model.

Parameters:

index The index of the datapoint in the model.

Returns:

The brush to use for painting.

Definition at line 816 of file KDChartAbstractDiagram.cpp.

```
822                               :
QRect AbstractDiagram::visualRect(const QModelIndex &) const
```

7.7.3.6 QBrush AbstractDiagram::brush (int *dataset*) const

Retrieve the brush to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the brush for.

Returns:

The brush to use for painting.

Definition at line 808 of file KDChartAbstractDiagram.cpp.

```
815 {
```

7.7.3.7 QBrush AbstractDiagram::brush () const

Retrieve the brush to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:
The brush to use for painting.

Definition at line 802 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), and paintMarker().

```
807 {
```

7.7.3.8 virtual const QPair< QPointF, QPointF > KDChart::AbstractDiagram::calculateDataBoundaries () const `[protected, pure virtual]`

Implemented in KDChart::BarDiagram, KDChart::LineDiagram, KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

Referenced by dataBoundaries().

7.7.3.9 bool AbstractDiagram::checkInvariants (bool *justReturnTheStatus* = false) const `[protected, virtual]`

Definition at line 930 of file KDChartAbstractDiagram.cpp.

References coordinatePlane().

Referenced by KDChart::RingDiagram::calculateDataBoundaries(), KDChart::PolarDiagram::calculateDataBoundaries(), KDChart::PieDiagram::calculateDataBoundaries(), KDChart::LineDiagram::calculateDataBoundaries(), KDChart::BarDiagram::calculateDataBoundaries(), KDChart::RingDiagram::paint(), KDChart::PolarDiagram::paint(), KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and paintMarker().

```
930                                {
931         Q_ASSERT_X ( model(), "AbstractDiagram::checkInvariants()",
932                      "There is no usable model set, for the diagram." );
933 
934         Q_ASSERT_X ( coordinatePlane(), "AbstractDiagram::checkInvariants()",
935                      "There is no usable coordinate plane set, for the diagram." );
936     }
937     return model() && coordinatePlane();
938 }
939 
940 int AbstractDiagram::datasetDimension( ) const
```

7.7.3.10 QModelIndex AbstractDiagram::columnToIndex (int *column*) const `[protected]`

Definition at line 317 of file KDChartAbstractDiagram.cpp.

```
323 {
```

7.7.3.11 bool AbstractDiagram::compare (const AbstractDiagram ∗ *other*) const

Returns true if both diagrams have the same settings.

Definition at line 135 of file KDChartAbstractDiagram.cpp.

```
{
    if( other == this ) return true;
    if( ! other ){
        //qDebug() << "AbstractDiagram::compare() cannot compare to Null pointer";
        return false;
    }
    /*
    qDebug() << "\n             AbstractDiagram::compare() QAbstractScrollArea:";
            // compare QAbstractScrollArea properties
    qDebug() <<
            ((horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
            (verticalScrollBarPolicy()    == other->verticalScrollBarPolicy()));
    qDebug() << "AbstractDiagram::compare() QFrame:";
            // compare QFrame properties
    qDebug() <<
            ((frameShadow() == other->frameShadow()) &&
            (frameShape()   == other->frameShape()) &&
            (frameWidth()   == other->frameWidth()) &&
            (lineWidth()    == other->lineWidth()) &&
            (midLineWidth() == other->midLineWidth()));
    qDebug() << "AbstractDiagram::compare() QAbstractItemView:";
            // compare QAbstractItemView properties
    qDebug() <<
            ((alternatingRowColors() == other->alternatingRowColors()) &&
            (hasAutoScroll()         == other->hasAutoScroll()) &&
#if QT_VERSION > 0x040199
            (dragDropMode()          == other->dragDropMode()) &&
            (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
            (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
            (verticalScrollMode()    == other->verticalScrollMode()) &&
#endif
            (dragEnabled()           == other->dragEnabled()) &&
            (editTriggers()          == other->editTriggers()) &&
            (iconSize()              == other->iconSize()) &&
            (selectionBehavior()     == other->selectionBehavior()) &&
            (selectionMode()         == other->selectionMode()) &&
            (showDropIndicator()     == other->showDropIndicator()) &&
            (tabKeyNavigation()      == other->tabKeyNavigation()) &&
            (textElideMode()         == other->textElideMode()));
    qDebug() << "AbstractDiagram::compare() AttributesModel: ";
            // compare all of the properties stored in the attributes model
    qDebug() << attributesModel()->compare( other->attributesModel() );
    qDebug() << "AbstractDiagram::compare() own:";
            // compare own properties
    qDebug() <<
            ((rootIndex().column()            == other->rootIndex().column()) &&
            (rootIndex().row()                == other->rootIndex().row()) &&
            (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
            (antiAliasing()                   == other->antiAliasing()) &&
            (percentMode()                    == other->percentMode()) &&
            (datasetDimension()               == other->datasetDimension()));
    */
    return  // compare QAbstractScrollArea properties
            (horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
            (verticalScrollBarPolicy()   == other->verticalScrollBarPolicy()) &&
            // compare QFrame properties
            (frameShadow()  == other->frameShadow()) &&
            (frameShape()   == other->frameShape()) &&
            (frameWidth()   == other->frameWidth()) &&
            (lineWidth()    == other->lineWidth()) &&
            (midLineWidth() == other->midLineWidth()) &&
            // compare QAbstractItemView properties
            (alternatingRowColors()  == other->alternatingRowColors()) &&
            (hasAutoScroll()         == other->hasAutoScroll()) &&
#if QT_VERSION > 0x040199
            (dragDropMode()          == other->dragDropMode()) &&
            (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
            (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
            (verticalScrollMode()    == other->verticalScrollMode()) &&
#endif
            (dragEnabled()           == other->dragEnabled()) &&
            (editTriggers()          == other->editTriggers()) &&
            (iconSize()              == other->iconSize()) &&
            (selectionBehavior()     == other->selectionBehavior()) &&
            (selectionMode()         == other->selectionMode()) &&
            (showDropIndicator()     == other->showDropIndicator()) &&
            (tabKeyNavigation()      == other->tabKeyNavigation()) &&
            (textElideMode()         == other->textElideMode()) &&
            // compare all of the properties stored in the attributes model
            attributesModel()->compare( other->attributesModel() ) &&
            // compare own properties
            (rootIndex().column()             == other->rootIndex().column()) &&
            (rootIndex().row()                == other->rootIndex().row()) &&
            (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
            (antiAliasing()                   == other->antiAliasing()) &&
            (percentMode()                    == other->percentMode()) &&
            (datasetDimension()               == other->datasetDimension());
}
```

7.7.3.12 AbstractCoordinatePlane ∗ AbstractDiagram::coordinatePlane () const

The coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. By default this is a CartesianCoordinatePlane.

Returns:

The coordinate plane associated with the diagram.

Definition at line 226 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by checkInvariants(), KDChart::AbstractCartesianDiagram::layoutPlanes(), KDChart::PolarDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), KDChart::AbstractPolarDiagram::polarCoordinatePlane(), and KDChart::AbstractCartesianDiagram::setCoordinatePlane().

```
{
    return d->plane;
}
```

7.7.3.13 const QPair< QPointF, QPointF > AbstractDiagram::dataBoundaries () const

Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

This method returns a chached result of calculations done by calculateDataBoundaries. Classes derived from AbstractDiagram must implement the calculateDataBoundaries function, to specify their own way of calculating the data boundaries. If derived classes want to force recalculation of the data boundaries, they can call setDataBoundariesDirty()

Returned value is in diagram coordinates.

Definition at line 231 of file KDChartAbstractDiagram.cpp.

References calculateDataBoundaries(), and d.

Referenced by KDChart::CartesianCoordinatePlane::getRawDataBoundingRectFromDiagrams(), KDChart::PolarCoordinatePlane::layoutDiagrams(), KDChart::LineDiagram::paint(), and KDChart::BarDiagram::paint().

```
{
    if( d->databoundariesDirty ){
        d->databoundaries = calculateDataBoundaries ();
        d->databoundariesDirty = false;
    }
    return d->databoundaries;
}
```

7.7.3.14 void AbstractDiagram::dataChanged (const QModelIndex & *topLeft*, const QModelIndex & *bottomRight*) `[virtual]`

[reimplemented]

Definition at line 338 of file KDChartAbstractDiagram.cpp.

References d.

```
{
  // We are still too dumb to do intelligent updates...
  d->databoundariesDirty = true;
  scheduleDelayedItemsLayout();
}


```

7.7.3.15 void KDChart::AbstractDiagram::dataHidden () `[protected]`

This signal is emitted, when the hidden status of at least one data cell was (un)set.

7.7.3.16 QList< QBrush > AbstractDiagram::datasetBrushes () const

The set of dataset brushes currently used, for use in legends, etc.

Note:

Cell-level override brushes, if set, take precedence over the dataset values, so you might need to check these too, in order to find the brush, that is used for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 894 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), KDChart::Legend::datasetCount(), and KDChart::Legend::setBrushesFromDiagram().

```
                                                                                  {
        QBrush brush = qVariantValue<QBrush>( attributesModel()->headerData( i, Qt::Vertical, DatasetB
        ret << brush;
    }

    return ret;
}

QList<QPen> AbstractDiagram::datasetPens() const
```

7.7.3.17 int AbstractDiagram::datasetDimension () const

The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

For each dimension it will expect one column of values in the model. If the dimensionality is 1, automatic values will be used for the abscissa.

For example a diagram with the default dimension of 1, will have one column per datapoint (the y values) and will use automatic values for the x axis (1, 2, 3, ... n). If the dimension is 2, the diagram will use the first, (and the third, fifth, etc) columns as X values, and the second, (and the fourth, sixth, etc) column as Y values.

Returns:

The dataset dimension of the diagram.

Definition at line 942 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::calculateDataBoundaries(), KDChart::LineDiagram::getCellValues(), KDChart::CartesianCoordinatePlane::getDataDimensionsList(), KDChart::LineDiagram::paint(), and KDChart::LineDiagram::setType().

```
{
```

7.7.3.18 QStringList AbstractDiagram::datasetLabels () const

The set of dataset labels currently displayed, for use in legends, etc.

Returns:

The set of dataset labels currently displayed.

Definition at line 882 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), and KDChart::Legend::datasetCount().

```
                                                 : " << attributesModel()->columnCount(attributesModel
    const int columnCount = attributesModel()->columnCount(attributesModelRootIndex());
    for( int i = datasetDimension()-1; i < columnCount; i += datasetDimension() ){
        //qDebug() << "dataset label: " << attributesModel()->headerData( i, Qt::Horizontal, Qt::Displ
        ret << attributesModel()->headerData( i, Qt::Horizontal, Qt::DisplayRole ).toString();
    }
    return ret;
}

QList<QBrush> AbstractDiagram::datasetBrushes() const
```

7.7.3.19 QList< MarkerAttributes > AbstractDiagram::datasetMarkers () const

The set of dataset markers currently used, for use in legends, etc.

Note:

Cell-level override markers, if set, take precedence over the dataset values, so you might need to check these too, in order to find the marker, that is shown for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 917 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
919                                                                                      {
920             DataValueAttributes a =
921                 qVariantValue<DataValueAttributes>( attributesModel()->headerData( i, Qt::Vertical, DataVa
922             const MarkerAttributes &ma = a.markerAttributes();
923             ret << ma;
924         }
925         return ret;
926 }
927
928 bool AbstractDiagram::checkInvariants( bool justReturnTheStatus ) const
```

7.7.3.20 QList< QPen > AbstractDiagram::datasetPens () const

The set of dataset pens currently used, for use in legends, etc.

Note:

Cell-level override pens, if set, take precedence over the dataset values, so you might need to check these too, in order to find the pens, that is used for a single cell.

Returns:

The current set of dataset pens.

Definition at line 906 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
908                                                                                      {
909             QPen pen = qVariantValue<QPen>( attributesModel()->headerData( i, Qt::Vertical, DatasetPenRole
910             ret << pen;
911         }
912         return ret;
913 }
914
915 QList<MarkerAttributes> AbstractDiagram::datasetMarkers() const
```

7.7.3.21 DataValueAttributes AbstractDiagram::dataValueAttributes (const QModelIndex & *index*) const

Retrieve the DataValueAttributes for the given index.

This will fall back automatically to what was set at dataset or model level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the attributes for.

Returns:

The DataValueAttributes for the given index.

Definition at line 427 of file KDChartAbstractDiagram.cpp.

```
433 {
```

7.7.3.22 DataValueAttributes AbstractDiagram::dataValueAttributes (int *column*) const

Retrieve the DataValueAttributes for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the attributes for.

Returns:

The DataValueAttributes for the given dataset.

Definition at line 420 of file KDChartAbstractDiagram.cpp.

```
426 {
```

7.7.3.23 DataValueAttributes AbstractDiagram::dataValueAttributes () const

Retrieve the DataValueAttributes speficied globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The global DataValueAttributes.

Definition at line 414 of file KDChartAbstractDiagram.cpp.

Referenced by paintDataValueText(), and paintMarker().

```
419 {
```

7.7.3.24 void AbstractDiagram::doItemsLayout () `[virtual]`

[reimplemented]

Definition at line 329 of file KDChartAbstractDiagram.cpp.

References d, and update().

```
329                      {
330         d->plane->layoutDiagrams();
331         update();
332     }
333     QAbstractItemView::doItemsLayout();
334 }
335
336 void AbstractDiagram::dataChanged( const QModelIndex &topLeft,
```

7.7.3.25 int AbstractDiagram::horizontalOffset () const `[virtual]`

[reimplemented]

Definition at line 839 of file KDChartAbstractDiagram.cpp.

```
841 { return 0; }
```

7.7.3.26 QModelIndex AbstractDiagram::indexAt (const QPoint & *point*) const `[virtual]`

[reimplemented]

Definition at line 833 of file KDChartAbstractDiagram.cpp.

```
835 { return QModelIndex(); }
```

7.7.3.27 bool AbstractDiagram::isHidden (const QModelIndex & *index*) const

Retrieve the hidden status for the given index.

This will fall back automatically to what was set at dataset or diagram level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the hidden status for.

Returns:

The hidden status for the given index.

Definition at line 386 of file KDChartAbstractDiagram.cpp.

7.7.3.28 bool AbstractDiagram::isHidden (int *column*) const

Retrieve the hidden status for the given dataset.

This will fall back automatically to what was set at diagram level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the hidden status for.

Returns:

The hidden status for the given dataset.

Definition at line 379 of file KDChartAbstractDiagram.cpp.

```
385 {
```

7.7.3.29 bool AbstractDiagram::isHidden () const

Retrieve the hidden status speficied globally.

This will fall back automatically to the default settings (= not hidden), if there are no specific settings.

Returns:
The global hidden status.

Definition at line 373 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), KDChart::LineDiagram::paint(), and KDChart::LineDiagram::valueForCellTesting().

```
{
```

7.7.3.30 bool AbstractDiagram::isIndexHidden (const QModelIndex & *index*) const `[virtual]`

[reimplemented]

Definition at line 845 of file KDChartAbstractDiagram.cpp.

```
{}
```

7.7.3.31 QStringList AbstractDiagram::itemRowLabels () const

The set of item row labels currently displayed, for use in Abscissa axes, etc.

Returns:
The set of item row labels currently displayed.

Definition at line 870 of file KDChartAbstractDiagram.cpp.

```
                                                 : " << attributesModel()->rowCount(attributesModelRoo
    const int rowCount = attributesModel()->rowCount(attributesModelRootIndex());
    for( int i = 0; i < rowCount; ++i ){
        //qDebug() << "item row label: " << attributesModel()->headerData( i, Qt::Vertical, Qt::Displa
        ret << attributesModel()->headerData( i, Qt::Vertical, Qt::DisplayRole ).toString();
    }
    return ret;
}

QStringList AbstractDiagram::datasetLabels() const
```

7.7.3.32 void KDChart::AbstractDiagram::modelsChanged () `[protected]`

This signal is emitted, when either the model or the AttributesModel is replaced.

Referenced by setAttributesModel(), and setModel().

7.7.3.33 QModelIndex AbstractDiagram::moveCursor (CursorAction *cursorAction*, Qt::KeyboardModifiers *modifiers*) `[virtual]`

[reimplemented]

Definition at line 836 of file KDChartAbstractDiagram.cpp.

```
{ return 0; }
```

7.7.3.34 virtual void KDChart::AbstractDiagram::paint (PaintContext ∗ *paintContext*) `[pure virtual]`

Draw the diagram contents to the rectangle and painter, that are passed in as part of the paint context.

Parameters:

paintContext All information needed for painting.

Implemented in KDChart::BarDiagram, KDChart::LineDiagram, KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

7.7.3.35 void AbstractDiagram::paintDataValueText (QPainter ∗ *painter*, const QModelIndex & *index*, const QPointF & *pos*, double *value*)

Definition at line 474 of file KDChartAbstractDiagram.cpp.

References KDChart::RelativePosition::alignment(), KDChart::TextAttributes::calculatedFont(), d, KDChart::DataValueAttributes::dataLabel(), dataValueAttributes(), KDChart::DataValueAttributes::decimalDigits(), KDChart::TextAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::TextAttributes::pen(), KDChart::DataValueAttributes::position(), KDChart::DataValueAttributes::prefix(), KDChart::TextAttributes::rotation(), KDChart::DataValueAttributes::showRepetitiveDataLabels(), KDChart::DataValueAttributes::suffix(), and KDChart::DataValueAttributes::textAttributes().

Referenced by KDChart::RingDiagram::paint(), and KDChart::PolarDiagram::paint().

```
{
    // paint one data series
    const DataValueAttributes a( dataValueAttributes(index) );
    if ( !a.isVisible() ) return;

    // handle decimal digits
    int decimalDigits = a.decimalDigits();
    int decimalPos = QString::number(  value ).indexOf( QLatin1Char( '.' ) );
    QString roundedValue;
    if ( a.dataLabel().isNull() ) {
        if ( decimalPos > 0 && value != 0 )
            roundedValue =  roundValues ( value, decimalPos, decimalDigits );
        else
            roundedValue = QString::number(  value );
    } else
        roundedValue = a.dataLabel();
        // handle prefix and suffix
    if ( !a.prefix().isNull() )
        roundedValue.prepend( a.prefix() );

    if ( !a.suffix().isNull() )
        roundedValue.append( a.suffix() );

    const TextAttributes ta( a.textAttributes() );
    // FIXME draw the non-text bits, background, etc
    if ( ta.isVisible() ) {

        QPointF pt( pos );
        /* for debugging:
        PainterSaver painterSaver( painter );
        painter->setPen( Qt::black );
        painter->drawLine( pos - QPointF( 1,1), pos + QPointF( 1,1) );
        painter->drawLine( pos - QPointF(-1,1), pos + QPointF(-1,1) );
        */

        // adjust the text start point position, if alignment is not Bottom/Left
        const RelativePosition relPos( a.position( value >= 0.0 ) );
        const Qt::Alignment alignBottomLeft = Qt::AlignBottom | Qt::AlignLeft;
        const QFont calculatedFont( ta.calculatedFont( d->plane, KDChartEnums::MeasureOrientationMinim
        //qDebug() << "calculatedFont's point size:" << calculatedFont.pointSizeF();
        if( (relPos.alignment() & alignBottomLeft) != alignBottomLeft ){
            const QRectF boundRect(
                    d->cachedFontMetrics( calculatedFont, this )->boundingRect( roundedValue ) );
            if( relPos.alignment() & Qt::AlignRight )
                pt.rx() -= boundRect.width();
            else if( relPos.alignment() & Qt::AlignHCenter )
                pt.rx() -= 0.5 * boundRect.width();

            if( relPos.alignment() & Qt::AlignTop )
                pt.ry() += boundRect.height();
            else if( relPos.alignment() & Qt::AlignVCenter )
                pt.ry() += 0.5 * boundRect.height();
        }

        // FIXME draw the non-text bits, background, etc

        if ( a.showRepetitiveDataLabels() ||
             pos.x() <= d->lastX ||
             d->lastRoundedValue != roundedValue ) {
            d->lastRoundedValue = roundedValue;
            d->lastX = pos.x();

            PainterSaver painterSaver( painter );
            painter->setPen( ta.pen() );
            painter->setFont( calculatedFont );
            painter->translate( pt );
            painter->rotate( ta.rotation() );
            painter->drawText( QPointF(0, 0), roundedValue );
        }
    }
}


```

7.7.3.36 void AbstractDiagram::paintDataValueTexts (QPainter ∗ *painter*) [protected, virtual]

Definition at line 576 of file KDChartAbstractDiagram.cpp.

```
                                                                               {
       for ( int j=0; j< rowCount; ++j ) {
           const QModelIndex index = model()->index( j, i, rootIndex() );
           double value = model()->data( index ).toDouble();
           const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
           paintDataValueText( painter, index, pos, value );
       }
```

```
    }
}


```

7.7.3.37 void AbstractDiagram::paintMarker (QPainter ∗ *painter*, const QModelIndex & *index*, const QPointF & *pos*)

Definition at line 592 of file KDChartAbstractDiagram.cpp.

References brush(), checkInvariants(), dataValueAttributes(), KDChart::MarkerAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::DataValueAttributes::markerAttributes(), KDChart::MarkerAttributes::markerColor(), KDChart::MarkerAttributes::markerSize(), paintMarker(), and KDChart::MarkerAttributes::pen().

```
{

    if ( !checkInvariants() ) return;
    DataValueAttributes a = dataValueAttributes(index);
    if ( !a.isVisible() ) return;
    const MarkerAttributes &ma = a.markerAttributes();
    if ( !ma.isVisible() ) return;

    PainterSaver painterSaver( painter );
    QSizeF maSize( ma.markerSize() );
    QBrush indexBrush( brush( index ) );
    QPen indexPen( ma.pen() );
    if ( ma.markerColor().isValid() )
        indexBrush.setColor( ma.markerColor() );

    paintMarker( painter, ma, indexBrush, indexPen, pos, maSize );
}


```

7.7.3.38 void AbstractDiagram::paintMarker (QPainter ∗ *painter*, const MarkerAttributes & *markerAttributes*, const QBrush & *brush*, const QPen &, const QPointF & *point*, const QSizeF & *size*) `[virtual]`

Definition at line 614 of file KDChartAbstractDiagram.cpp.

References KDChart::MarkerAttributes::markerStyle().

Referenced by KDChart::MarkerLayoutItem::paintIntoRect(), and paintMarker().

```
{

    const QPen oldPen( painter->pen() );
    // Pen is used to paint 4Pixels - 1 Pixel - Ring and FastCross types.
    // make sure to use the brush color - see above in those cases.
    const bool isFourPixels = (markerAttributes.markerStyle() == MarkerAttributes::Marker4Pixels);
    if( isFourPixels || (markerAttributes.markerStyle() == MarkerAttributes::Marker1Pixel) ){
        // for high-performance point charts with tiny point markers:
        painter->setPen( QPen( brush.color().light() ) );
        if( isFourPixels ){
            const qreal x = pos.x();
            const qreal y = pos.y();
            painter->drawLine( QPointF(x-1.0,y-1.0),
                               QPointF(x+1.0,y-1.0) );
            painter->drawLine( QPointF(x-1.0,y),
                               QPointF(x+1.0,y) );
            painter->drawLine( QPointF(x-1.0,y+1.0),
                               QPointF(x+1.0,y+1.0) );
        }
        painter->drawPoint( pos );
    }else{
        PainterSaver painterSaver( painter );
        // we only a solid line surrounding the markers
        QPen painterPen( pen );
        painterPen.setStyle( Qt::SolidLine );
        painter->setPen( painterPen );
        painter->setBrush( brush );
        painter->setRenderHint ( QPainter::Antialiasing );
        painter->translate( pos );
        switch ( markerAttributes.markerStyle() ) {
            case MarkerAttributes::MarkerCircle:
                painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                            maSize.height(), maSize.width()) );
                break;
            case MarkerAttributes::MarkerSquare:
                {
                    QRectF rect( 0 - maSize.width()/2, 0 - maSize.height()/2,
                                maSize.width(), maSize.height() );
                    painter->drawRect( rect );
                    painter->fillRect( rect, brush.color() );
                    break;
                }
            case MarkerAttributes::MarkerDiamond:
                {
                    QVector <QPointF > diamondPoints;
                    QPointF top, left, bottom, right;
                    top    = QPointF( 0, 0 - maSize.height()/2 );
                    left   = QPointF( 0 - maSize.width()/2, 0 );
                    bottom = QPointF( 0, maSize.height()/2 );
                    right  = QPointF( maSize.width()/2, 0 );
                    diamondPoints << top << left << bottom << right;
                    painter->drawPolygon( diamondPoints );
                    break;
                }
            // both handled on top of the method:
            case MarkerAttributes::Marker1Pixel:
            case MarkerAttributes::Marker4Pixels:
                    break;
            case MarkerAttributes::MarkerRing:
                {
                    painter->setPen( QPen( brush.color() ) );
                    painter->setBrush( Qt::NoBrush );
                    painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                                        maSize.height(), maSize.width()) );
                    break;
                }
            case MarkerAttributes::MarkerCross:
                {
                    QRectF rect( maSize.width()*-0.5, maSize.height()*-0.2,
                                 maSize.width(), maSize.height()*0.4 );
                    painter->drawRect( rect );
                    rect.setTopLeft(QPointF( maSize.width()*-0.2, maSize.height()*-0.5 ));
                    rect.setSize(QSizeF( maSize.width()*0.4, maSize.height() ));
                    painter->drawRect( rect );
                    break;
                }
            case MarkerAttributes::MarkerFastCross:
                {
                    QPointF left, right, top, bottom;
                    left  = QPointF( -maSize.width()/2, 0 );
                    right = QPointF( maSize.width()/2, 0 );
                    top   = QPointF( 0, -maSize.height()/2 );
                    bottom= QPointF( 0, maSize.height()/2 );
                    painter->setPen( QPen( brush.color() ) );
                    painter->drawLine( left, right );
                    painter->drawLine(  top, bottom );
                    break;
                }
            default:
                Q_ASSERT_X ( false, "paintMarkers()",
                            "Type item does not match a defined Marker Type." );
        }
    }
    painter->setPen( oldPen );
}

void AbstractDiagram::paintMarkers( QPainter* painter )
```

7.7.3.39 void AbstractDiagram::paintMarkers (QPainter ∗ *painter*) [protected, virtual]

Definition at line 716 of file KDChartAbstractDiagram.cpp.

```
                                                                               {
       for ( int j=0; j< rowCount; ++j ) {
           const QModelIndex index = model()->index( j, i, rootIndex() );
           double value = model()->data( index ).toDouble();
           const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
           paintMarker( painter, index, pos );
       }
    }
}


```

7.7.3.40 QPen AbstractDiagram::pen (const QModelIndex & *index*) const

Retrieve the pen to be used, for painting the datapoint at the given index in the model.

Parameters:

index The index of the datapoint in the model.

Returns:

The pen to use for painting.

Definition at line 770 of file KDChartAbstractDiagram.cpp.

```
{
```

7.7.3.41 QPen AbstractDiagram::pen (int *dataset*) const

Retrieve the pen to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the pen for.

Returns:

The pen to use for painting.

Definition at line 762 of file KDChartAbstractDiagram.cpp.

```
769 {
```

7.7.3.42 QPen AbstractDiagram::pen () const

Retrieve the pen to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The pen to use for painting.

Definition at line 756 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::PieDiagram::paint(), and KDChart::LineDiagram::paint().

```
761 {
```

7.7.3.43 bool AbstractDiagram::percentMode () const

Definition at line 468 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::CartesianCoordinatePlane::getDataDimensionsList().

7.7.3.44 void KDChart::AbstractDiagram::propertiesChanged () `[protected]`

Emitted upon change of a property of the Diagram.

Referenced by KDChart::LineDiagram::resetLineAttributes(), setDataValueAttributes(), KDChart::LineDiagram::setLineAttributes(), KDChart::LineDiagram::setThreeDLineAttributes(), and KDChart::LineDiagram::setType().

7.7.3.45 virtual void KDChart::AbstractDiagram::resize (const QSizeF & *area*) `[pure virtual]`

Called by the widget's sizeEvent.

Adjust all internal structures, that are calculated, dependending on the size of the widget.

Parameters:

area

Implemented in KDChart::BarDiagram, KDChart::LineDiagram, KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

7.7.3.46 void AbstractDiagram::scrollTo (const QModelIndex & *index*, ScrollHint *hint* = EnsureVisible) `[virtual]`

[reimplemented]

Definition at line 830 of file KDChartAbstractDiagram.cpp.

```
832 { return QModelIndex(); }
```

7.7.3.47 void AbstractDiagram::setAllowOverlappingDataValueTexts (bool *allow*)

Set whether data value labels are allowed to overlap.

Parameters:

allow True means that overlapping labels are allowed.

Definition at line 440 of file KDChartAbstractDiagram.cpp.

References d.

```
445 {
```

7.7.3.48 void AbstractDiagram::setAntiAliasing (bool *enabled*)

Set whether anti-aliasing is to be used while rendering this diagram.

Parameters:

enabled True means that AA is enabled.

Definition at line 451 of file KDChartAbstractDiagram.cpp.

References d.

```
456 {
```

7.7.3.49 void AbstractDiagram::setAttributesModel (AttributesModel ∗ *model*) `[virtual]`

Associate an AttributesModel with this diagram.

Note that the diagram does _not_ take ownership of the AttributesModel. This should thus only be used with AttributesModels that have been explicitely created by the user, and are owned by her. Setting an AttributesModel that is internal to another diagram is an error.

Correct:

```
AttributesModel *am = new AttributesModel( model, 0 );
diagram1->setAttributesModel( am );
diagram2->setAttributesModel( am );
```

Wrong:

```
diagram1->setAttributesModel( diagram2->attributesModel() );
```

Parameters:

model The AttributesModel to use for this diagram.

See also:

AttributesModel, usesExternalAttributesModel

Definition at line 261 of file KDChartAbstractDiagram.cpp.

References d, and modelsChanged().

```
{
    if( amodel->sourceModel() != model() ) {
        qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
                 "Trying to set an attributesmodel which works on a different "
                 "model than the diagram.");
        return;
    }
    if( qobject_cast<PrivateAttributesModel*>(amodel) ) {
        qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
                 "Trying to set an attributesmodel that is private to another diagram.");
        return;
    }
    d->setAttributesModel(amodel);
    scheduleDelayedItemsLayout();
    d->databoundariesDirty = true;
    emit modelsChanged();
}
```

7.7.3.50 void AbstractDiagram::setAttributesModelRootIndex (const QModelIndex & *idx*) `[protected]`

Definition at line 301 of file KDChartAbstractDiagram.cpp.

References d.

7.7.3.51 void AbstractDiagram::setBrush (const QBrush & *brush*)

Set the brush to be used, for painting all datasets in the model.

Parameters:

brush The brush to use.

Definition at line 786 of file KDChartAbstractDiagram.cpp.

```
{
```

7.7.3.52 void AbstractDiagram::setBrush (int *dataset*, const QBrush & *brush*)

Set the brush to be used, for painting the given dataset.

Parameters:

dataset The dataset’s column in the model.

pen The brush to use.

Definition at line 793 of file KDChartAbstractDiagram.cpp.

```
801 {
```

7.7.3.53 void AbstractDiagram::setBrush (const QModelIndex & *index*, const QBrush & *brush*)

Set the brush to be used, for painting the datapoint at the given index.

Parameters:

index The datapoint's index in the model.

brush The brush to use.

Definition at line 778 of file KDChartAbstractDiagram.cpp.

```
785 {
```

7.7.3.54 void AbstractDiagram::setCoordinatePlane (AbstractCoordinatePlane ∗ *plane*) `[virtual]`

Set the coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. The chart takes ownership.

Returns:

The coordinate plane associated with the diagram.

Reimplemented in KDChart::AbstractCartesianDiagram.

Definition at line 324 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractCoordinatePlane::addDiagram(), KDChart::AbstractCartesianDiagram::setCoordinatePlane(), and KDChart::AbstractCoordinatePlane::takeDiagram().

```
328 {
```

7.7.3.55 void AbstractDiagram::setDataBoundariesDirty () const `[protected]`

Definition at line 240 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::BarDiagram::setThreeDBarAttributes(), KDChart::LineDiagram::setThreeDLineAttributes(), KDChart::LineDiagram::setType(), and KDChart::BarDiagram::setType().

```
241 {
242     d->databoundariesDirty = true;
243 }
```

7.7.3.56 void AbstractDiagram::setDatasetDimension (int *dimension*)

Sets the dataset dimension of the diagram.

See also:
datasetDimension.

Parameters:
dimension

Definition at line 947 of file KDChartAbstractDiagram.cpp.

References d.

```
954 {
```

7.7.3.57 void AbstractDiagram::setDataValueAttributes (const DataValueAttributes & *a*)

Set the DataValueAttributes for all datapoints in the model.

Parameters:
a The attributes to set.

Definition at line 434 of file KDChartAbstractDiagram.cpp.

References d.

```
439 {
```

7.7.3.58 void AbstractDiagram::setDataValueAttributes (int *dataset*, const DataValueAttributes & *a*)

Set the DataValueAttributes for the given dataset.

Parameters:
dataset The dataset to set the attributes for.

a The attributes to set.

Definition at line 406 of file KDChartAbstractDiagram.cpp.

References d.

```
413 {
```

7.7.3.59 void AbstractDiagram::setDataValueAttributes (const QModelIndex & *index*, const DataValueAttributes & *a*)

Set the DataValueAttributes for the given index.

Parameters:

index The datapoint to set the attributes for.

a The attributes to set.

Definition at line 395 of file KDChartAbstractDiagram.cpp.

References d, KDChart::DataValueLabelAttributesRole, and propertiesChanged().

```
395 {
396     d->attributesModel->setData(
397         d->attributesModel->mapFromSource( index ),
398         qVariantFromValue( a ),
399         DataValueLabelAttributesRole );
400     emit propertiesChanged();
401 }
402
403
```

7.7.3.60 void AbstractDiagram::setHidden (bool *hidden*)

Hide (or unhide, resp.) all datapoints in the model.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

hidden The hidden status to set.

Definition at line 365 of file KDChartAbstractDiagram.cpp.

References d.

```
372 {
```

7.7.3.61 void AbstractDiagram::setHidden (int *column*, bool *hidden*)

Hide (or unhide, resp.) a dataset.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

dataset The dataset to set the hidden status for.

hidden The hidden status to set.

Definition at line 356 of file KDChartAbstractDiagram.cpp.

References d.

```
364 {
```

7.7.3.62 void AbstractDiagram::setHidden (const QModelIndex & *index*, bool *hidden*)

Hide (or unhide, resp.) a data cell.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

index The datapoint to set the hidden status for.

hidden The hidden status to set.

Definition at line 347 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::DataHiddenRole.

```
355 {
```

7.7.3.63 void AbstractDiagram::setModel (QAbstractItemModel ∗ *model*) `[virtual]`

Associate a model with the diagram.

Definition at line 245 of file KDChartAbstractDiagram.cpp.

References d, KDChart::AttributesModel::initFrom(), and modelsChanged().

```
246 {
247   QAbstractItemView::setModel( newModel );
248   AttributesModel* amodel = new PrivateAttributesModel( newModel, this );
249   amodel->initFrom( d->attributesModel );
250   d->setAttributesModel(amodel);
251   scheduleDelayedItemsLayout();
252   d->databoundariesDirty = true;
253   emit modelsChanged();
254 }
```

7.7.3.64 void AbstractDiagram::setPen (const QPen & *pen*)

Set the pen to be used, for painting all datasets in the model.

Parameters:

pen The pen to use.

Definition at line 740 of file KDChartAbstractDiagram.cpp.

```
746 {
```

7.7.3.65 void AbstractDiagram::setPen (int *dataset*, const QPen & *pen*)

Set the pen to be used, for painting the given dataset.

Parameters:

dataset The dataset's row in the model.

pen The pen to use.

Definition at line 747 of file KDChartAbstractDiagram.cpp.

```
755 {
```

7.7.3.66 void AbstractDiagram::setPen (const QModelIndex & *index*, const QPen & *pen*)

Set the pen to be used, for painting the datapoint at the given index.

Parameters:

index The datapoint's index in the model.

pen The pen to use.

Definition at line 732 of file KDChartAbstractDiagram.cpp.

```
739 {
```

7.7.3.67 void AbstractDiagram::setPercentMode (bool *percent*)

Definition at line 462 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::setType(), and KDChart::BarDiagram::setType().

```
467 {
```

7.7.3.68 void AbstractDiagram::setRootIndex (const QModelIndex & *idx*) `[virtual]`

Set the root index in the model, where the diagram starts referencing data for display.

[reimplemented]

Definition at line 294 of file KDChartAbstractDiagram.cpp.

References d.

7.7.3.69 void AbstractDiagram::setSelection (const QRect & *rect*, QItemSelectionModel::SelectionFlags *command*) `[virtual]`

[reimplemented]

Definition at line 848 of file KDChartAbstractDiagram.cpp.

```
850 { return QRegion(); }
```

7.7.3.70 void AbstractDiagram::update () const

Definition at line 961 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by doItemsLayout().

7.7.3.71 void KDChart::AbstractDiagram::useDefaultColors ()

Set the palette to be used, for painting datasets to the default palette.

See also:
KDChart::Palette. FIXME: fold into one usePalette (KDChart::Palette&) method

Definition at line 855 of file KDChartAbstractDiagram.cpp.

References d.

```
859 {
```

7.7.3.72 void KDChart::AbstractDiagram::useRainbowColors ()

Set the palette to be used, for painting datasets to the rainbow palette.

See also:
KDChart::Palette.

Definition at line 865 of file KDChartAbstractDiagram.cpp.

References d.

```
869 {
```

7.7.3.73 bool AbstractDiagram::usesExternalAttributesModel () const `[virtual]`

Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

See also:
setAttributesModel

Definition at line 280 of file KDChartAbstractDiagram.cpp.

References d.

```
281 {
282     return d->usesExternalAttributesModel();
283 }
```

7.7.3.74 void KDChart::AbstractDiagram::useSubduedColors ()

Set the palette to be used, for painting datasets to the subdued palette.

See also:
KDChart::Palette.

Definition at line 860 of file KDChartAbstractDiagram.cpp.

References d.

```
864 {
```

7.7.3.75 double AbstractDiagram::valueForCell (int *row*, int *column*) const `[protected]`

Helper method, retrieving the data value (DisplayRole) for a given row and column.

Parameters:
row The row to query.
column The column to query.

Returns:
The value of the display role at the given row and column as a double.

Definition at line 955 of file KDChartAbstractDiagram.cpp.

References attributesModelRootIndex(), and d.

Referenced by KDChart::LineDiagram::paint().

```
960 {
```

7.7.3.76 int AbstractDiagram::verticalOffset () const `[virtual]`

[reimplemented]

Definition at line 842 of file KDChartAbstractDiagram.cpp.

```
844 { return true; }
```

7.7.3.77 QRect AbstractDiagram::visualRect (const QModelIndex & *index*) const `[virtual]`

[reimplemented]

Definition at line 825 of file KDChartAbstractDiagram.cpp.

```
829 {}
```

7.7.3.78 QRegion AbstractDiagram::visualRegionForSelection (const QItemSelection & *selection*) const `[virtual]`

[reimplemented]

Definition at line 851 of file KDChartAbstractDiagram.cpp.

7.7.4 Member Data Documentation

7.7.4.1 Q_SIGNALS KDChart::AbstractDiagram::__pad0__ `[protected]`

Definition at line 589 of file KDChartAbstractDiagram.h.

The documentation for this class was generated from the following files:

- KDChartAbstractDiagram.h
- KDChartAbstractDiagram.cpp

7.8 KDChart::AbstractLayoutItem Class Reference

`#include <KDChartLayoutItems.h>`

Inheritance diagram for KDChart::AbstractLayoutItem:Collaboration diagram for KDChart::AbstractLayoutItem:

Public Member Functions

- AbstractLayoutItem (Qt::Alignment itemAlignment=0)
- virtual void paint (QPainter ∗)=0
- virtual void paintAll (QPainter &painter)

  Default impl: just call paint.

- virtual void paintCtx (PaintContext ∗context)

  Default impl: Paint the complete item using its layouted position and size.

- QLayout ∗ parentLayout ()
- void removeFromParentLayout ()
- void setParentLayout (QLayout ∗lay)
- virtual void setParentWidget (QWidget ∗widget)

  Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

- virtual void sizeHintChanged () const

  Report changed size hint: ask the parent widget to recalculate the layout.

Protected Attributes

- QWidget ∗ mParent
- QLayout ∗ mParentLayout

7.8.1 Constructor & Destructor Documentation

7.8.1.1 KDChart::AbstractLayoutItem::AbstractLayoutItem (Qt::Alignment *itemAlignment* = 0)

Definition at line 51 of file KDChartLayoutItems.h.

```
51                                                              :
52             QLayoutItem( itemAlignment ),
53             mParent( 0 ),
54             mParentLayout( 0 ) {}
```

7.8.2 Member Function Documentation

7.8.2.1 virtual void KDChart::AbstractLayoutItem::paint (QPainter ∗) `[pure virtual]`

Implemented in KDChart::CartesianAxis, KDChart::CartesianCoordinatePlane, KDChart::TextLayoutItem, KDChart::MarkerLayoutItem, KDChart::LineLayoutItem, KDChart::LineWithMarkerLayoutItem, KDChart::HorizontalLineLayoutItem, KDChart::VerticalLineLayoutItem, KDChart::AutoSpacerLayoutItem, and KDChart::PolarCoordinatePlane.

Referenced by KDChart::Legend::paint(), paintAll(), KDChart::AbstractArea::paintAll(), and paintCtx().

7.8.2.2 void KDChart::AbstractLayoutItem::paintAll (QPainter & *painter*) `[virtual]`

Default impl: just call paint.

Derived classes like KDChart::AbstractArea are providing additional action here.

Reimplemented in KDChart::AbstractArea, and KDChart::TextArea.

Definition at line 69 of file KDChartLayoutItems.cpp.

References paint().

```
70 {
71     paint( &painter );
72 }
```

7.8.2.3 void KDChart::AbstractLayoutItem::paintCtx (PaintContext ∗ *context*) `[virtual]`

Default impl: Paint the complete item using its layouted position and size.

Reimplemented in KDChart::CartesianAxis.

Definition at line 77 of file KDChartLayoutItems.cpp.

References paint(), and KDChart::PaintContext::painter().

```
78 {
79     if( context )
80         paint( context->painter() );
81 }
```

7.8.2.4 QLayout∗ KDChart::AbstractLayoutItem::parentLayout ()

Definition at line 74 of file KDChartLayoutItems.h.

```
75         {
76             return mParentLayout;
77         }
```

7.8.2.5 void KDChart::AbstractLayoutItem::removeFromParentLayout ()

Definition at line 78 of file KDChartLayoutItems.h.

Referenced by KDChart::Chart::takeCoordinatePlane().

```
79         {
80             if( mParentLayout ){
81                 if( widget() )
82                     mParentLayout->removeWidget( widget() );
83                 else
84                     mParentLayout->removeItem( this );
85             }
86         }
```

7.8.2.6 void KDChart::AbstractLayoutItem::setParentLayout (QLayout ∗ *lay*)

Definition at line 70 of file KDChartLayoutItems.h.

```
        {
            mParentLayout = lay;
        }
```

7.8.2.7 void KDChart::AbstractLayoutItem::setParentWidget (QWidget ∗ *widget*) `[virtual]`

Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

Thus, you need to call setParentWidget on every item, that has a non-fixed size.

Definition at line 64 of file KDChartLayoutItems.cpp.

References mParent.

Referenced by KDChart::Legend::buildLegend(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
{
    mParent = widget;
}
```

7.8.2.8 void KDChart::AbstractLayoutItem::sizeHintChanged () const `[virtual]`

Report changed size hint: ask the parent widget to recalculate the layout.

Definition at line 86 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextLayoutItem::sizeHint().

```
{
    // This is exactly like what QWidget::updateGeometry does.
//  qDebug("KDChart::AbstractLayoutItem::sizeHintChanged() called");
    if( mParent ) {
        if ( mParent->layout() )
            mParent->layout()->invalidate();
        else
            QApplication::postEvent( mParent, new QEvent( QEvent::LayoutRequest ) );
    }
}
```

7.8.3 Member Data Documentation

7.8.3.1 QWidget∗ KDChart::AbstractLayoutItem::mParent `[protected]`

Definition at line 88 of file KDChartLayoutItems.h.

Referenced by setParentWidget().

7.8.3.2 QLayout∗ KDChart::AbstractLayoutItem::mParentLayout `[protected]`

Definition at line 89 of file KDChartLayoutItems.h.

The documentation for this class was generated from the following files:

- KDChartLayoutItems.h
- KDChartLayoutItems.cpp

7.9 KDChart::AbstractPieDiagram Class Reference

`#include <KDChartAbstractPieDiagram.h>`

Inheritance diagram for KDChart::AbstractPieDiagram:Collaboration diagram for KDChart::AbstractPieDiagram:

Public Member Functions

- AbstractPieDiagram (QWidget ∗parent=0, PolarCoordinatePlane ∗plane=0)
- bool allowOverlappingDataValueTexts () const
- bool antiAliasing () const
- virtual AttributesModel ∗ attributesModel () const

  Returns the AttributesModel, that is used by this diagram.

- QBrush brush (const QModelIndex &index) const

  Retrieve the brush to be used, for painting the datapoint at the given index in the model.

- QBrush brush (int dataset) const

  Retrieve the brush to be used for the given dataset.

- QBrush brush () const

  Retrieve the brush to be used for painting datapoints globally.

- int columnCount () const
- bool compare (const AbstractDiagram ∗other) const

  Returns true if both diagrams have the same settings.

- AbstractCoordinatePlane ∗ coordinatePlane () const

  The coordinate plane associated with the diagram.

- const QPair< QPointF, QPointF > dataBoundaries () const

  Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

- virtual void dataChanged (const QModelIndex &topLeft, const QModelIndex &bottomRight)

  [reimplemented]

- QList< QBrush > datasetBrushes () const

  The set of dataset brushes currently used, for use in legends, etc.

- int datasetDimension () const

  The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

- QStringList datasetLabels () const

  The set of dataset labels currently displayed, for use in legends, etc.

- QList< MarkerAttributes > datasetMarkers () const

  The set of dataset markers currently used, for use in legends, etc.

- QList< QPen > datasetPens () const

  The set of dataset pens currently used, for use in legends, etc.

- DataValueAttributes dataValueAttributes (const QModelIndex &index) const

  Retrieve the DataValueAttributes for the given index.

- DataValueAttributes dataValueAttributes (int column) const

  Retrieve the DataValueAttributes for the given dataset.

- DataValueAttributes dataValueAttributes () const

  Retrieve the DataValueAttributes speficied globally.

- virtual void doItemsLayout ()

  [reimplemented]

- qreal granularity () const
- virtual int horizontalOffset () const

  [reimplemented]

- virtual QModelIndex indexAt (const QPoint &point) const

  [reimplemented]

- bool isHidden (const QModelIndex &index) const

  Retrieve the hidden status for the given index.

- bool isHidden (int column) const

  Retrieve the hidden status for the given dataset.

- bool isHidden () const

  Retrieve the hidden status speficied globally.

- virtual bool isIndexHidden (const QModelIndex &index) const

  [reimplemented]

- QStringList itemRowLabels () const

  The set of item row labels currently displayed, for use in Abscissa axes, etc.

- virtual QModelIndex moveCursor (CursorAction cursorAction, Qt::KeyboardModifiers modifiers)

  [reimplemented]

- virtual double numberOfGridRings () const=0
- virtual double numberOfValuesPerDataset () const=0
- virtual void paint (PaintContext ∗paintContext)=0

  Draw the diagram contents to the rectangle and painter, that are passed in as part of the paint context.

- void paintDataValueText (QPainter ∗painter, const QModelIndex &index, const QPointF &pos, double value)
- void paintMarker (QPainter ∗painter, const QModelIndex &index, const QPointF &pos)

- virtual void paintMarker (QPainter ∗painter, const MarkerAttributes &markerAttributes, const QBrush &brush, const QPen &, const QPointF &point, const QSizeF &size)
- QPen pen (const QModelIndex &index) const

  Retrieve the pen to be used, for painting the datapoint at the given index in the model.

- QPen pen (int dataset) const

  Retrieve the pen to be used for the given dataset.

- QPen pen () const

  Retrieve the pen to be used for painting datapoints globally.

- bool percentMode () const
- PieAttributes pieAttributes (const QModelIndex &index) const
- PieAttributes pieAttributes (int column) const
- PieAttributes pieAttributes () const
- const PolarCoordinatePlane ∗ polarCoordinatePlane () const
- virtual void resize (const QSizeF &area)=0

  Called by the widget's sizeEvent.

- virtual void scrollTo (const QModelIndex &index, ScrollHint hint=EnsureVisible)

  [reimplemented]

- void setAllowOverlappingDataValueTexts (bool allow)

  Set whether data value labels are allowed to overlap.

- void setAntiAliasing (bool enabled)

  Set whether anti-aliasing is to be used while rendering this diagram.

- virtual void setAttributesModel (AttributesModel ∗model)

  Associate an AttributesModel with this diagram.

- void setBrush (const QBrush &brush)

  Set the brush to be used, for painting all datasets in the model.

- void setBrush (int dataset, const QBrush &brush)

  Set the brush to be used, for painting the given dataset.

- void setBrush (const QModelIndex &index, const QBrush &brush)

  Set the brush to be used, for painting the datapoint at the given index.

- virtual void setCoordinatePlane (AbstractCoordinatePlane ∗plane)

  Set the coordinate plane associated with the diagram.

- void setDatasetDimension (int dimension)

  Sets the dataset dimension of the diagram.

- void setDataValueAttributes (const DataValueAttributes &a)

  Set the DataValueAttributes for all datapoints in the model.

- void setDataValueAttributes (int dataset, const DataValueAttributes &a)

Set the DataValueAttributes for the given dataset.

- void setDataValueAttributes (const QModelIndex &index, const DataValueAttributes &a)

  Set the DataValueAttributes for the given index.

- void setGranularity (qreal value)

  Set the granularity: the smaller the granularity the more your diagram segments will show facettes instead of rounded segments.

- void setHidden (bool hidden)

  Hide (or unhide, resp.) all datapoints in the model.

- void setHidden (int column, bool hidden)

  Hide (or unhide, resp.) a dataset.

- void setHidden (const QModelIndex &index, bool hidden)

  Hide (or unhide, resp.) a data cell.

- virtual void setModel (QAbstractItemModel ∗model)

  Associate a model with the diagram.

- void setPen (const QPen &pen)

  Set the pen to be used, for painting all datasets in the model.

- void setPen (int dataset, const QPen &pen)

  Set the pen to be used, for painting the given dataset.

- void setPen (const QModelIndex &index, const QPen &pen)

  Set the pen to be used, for painting the datapoint at the given index.

- void setPercentMode (bool percent)
- void setPieAttributes (int column, const PieAttributes &a)
- void setPieAttributes (const PieAttributes &a)
- virtual void setRootIndex (const QModelIndex &idx)

  Set the root index in the model, where the diagram starts referencing data for display.

- virtual void setSelection (const QRect &rect, QItemSelectionModel::SelectionFlags command)

  [reimplemented]

- void setStartPosition (int degrees)
- void setThreeDPieAttributes (const QModelIndex &index, const ThreeDPieAttributes &a)
- void setThreeDPieAttributes (int column, const ThreeDPieAttributes &a)
- void setThreeDPieAttributes (const ThreeDPieAttributes &a)
- int startPosition () const
- ThreeDPieAttributes threeDPieAttributes (const QModelIndex &index) const
- ThreeDPieAttributes threeDPieAttributes (int column) const
- ThreeDPieAttributes threeDPieAttributes () const
- void update () const
- void useDefaultColors ()

  Set the palette to be used, for painting datasets to the default palette.

- void useRainbowColors ()

  Set the palette to be used, for painting datasets to the rainbow palette.

- virtual bool usesExternalAttributesModel () const

  Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

- void useSubduedColors ()

  Set the palette to be used, for painting datasets to the subdued palette.

- virtual double valueTotals () const=0
- virtual int verticalOffset () const

  [reimplemented]

- virtual QRect visualRect (const QModelIndex &index) const

  [reimplemented]

- virtual QRegion visualRegionForSelection (const QItemSelection &selection) const

  [reimplemented]

- virtual ∼AbstractPieDiagram ()

Protected Member Functions

- QModelIndex attributesModelRootIndex () const
- virtual const QPair< QPointF, QPointF > calculateDataBoundaries () const=0
- virtual bool checkInvariants (bool justReturnTheStatus=false) const
- QModelIndex columnToIndex (int column) const
- void dataHidden ()

  This signal is emitted, when the hidden status of at least one data cell was (un)set.

- void modelsChanged ()

  This signal is emitted, when either the model or the AttributesModel is replaced.

- virtual void paintDataValueTexts (QPainter ∗painter)
- virtual void paintMarkers (QPainter ∗painter)
- void propertiesChanged ()

  Emitted upon change of a property of the Diagram.

- void setAttributesModelRootIndex (const QModelIndex &)
- void setDataBoundariesDirty () const
- double valueForCell (int row, int column) const

  Helper method, retrieving the data value (DisplayRole) for a given row and column.

Protected Attributes

- Q_SIGNALS __pad0__: void layoutChanged(AbstractDiagram∗)

7.9.1 Constructor & Destructor Documentation

7.9.1.1 AbstractPieDiagram::AbstractPieDiagram (QWidget ∗ *parent* = 0, PolarCoordinatePlane ∗ *plane* = 0) `[explicit]`

Definition at line 46 of file KDChartAbstractPieDiagram.cpp.

```
46                                                                                                       :
47     AbstractPolarDiagram( new Private(), parent, plane )
48 {
49     init();
50 }
```

7.9.1.2 AbstractPieDiagram::∼AbstractPieDiagram () `[virtual]`

Definition at line 52 of file KDChartAbstractPieDiagram.cpp.

```
53 {
54 }
```

7.9.2 Member Function Documentation

7.9.2.1 bool AbstractDiagram::allowOverlappingDataValueTexts () const `[inherited]`

Returns:

Whether data value labels are allowed to overlap.

Definition at line 446 of file KDChartAbstractDiagram.cpp.

References d.

```
450 {
```

7.9.2.2 bool AbstractDiagram::antiAliasing () const `[inherited]`

Returns:

Whether anti-aliasing is to be used for rendering this diagram.

Definition at line 457 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::paint().

```
461 {
```

7.9.2.3 AttributesModel ∗ AbstractDiagram::attributesModel () const `[virtual, inherited]`

Returns the AttributesModel, that is used by this diagram.

By default each diagram owns its own AttributesModel, which should never be deleted. Only if a user-supplied AttributesModel has been set does the pointer returned here not belong to the diagram.

Returns:

The AttributesModel associated with the diagram.

See also:

setAttributesModel

Definition at line 286 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::RingDiagram::paint(), KDChart::PolarDiagram::paint(), and KDChart::BarDiagram::setBarAttributes().

```
287 {
288     return d->attributesModel;
289 }
```

7.9.2.4 QModelIndex AbstractDiagram::attributesModelRootIndex () const `[protected, inherited]`

returns a QModelIndex pointing into the AttributesModel that corresponds to the root index of the diagram.

Definition at line 310 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::calculateDataBoundaries(), KDChart::BarDiagram::calculateDataBoundaries(), KDChart::LineDiagram::numberOfAbscissaSegments(), KDChart::BarDiagram::numberOfAbscissaSegments(), KDChart::LineDiagram::numberOfOrdinateSegments(), KDChart::BarDiagram::numberOfOrdinateSegments(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and KDChart::AbstractDiagram::valueForCell().

```
316 {
```

7.9.2.5 QBrush AbstractDiagram::brush (const QModelIndex & *index*) const `[inherited]`

Retrieve the brush to be used, for painting the datapoint at the given index in the model.

Parameters:

index The index of the datapoint in the model.

Returns:

The brush to use for painting.

Definition at line 816 of file KDChartAbstractDiagram.cpp.

```
822                                :
QRect AbstractDiagram::visualRect(const QModelIndex &) const
```

7.9.2.6 QBrush AbstractDiagram::brush (int *dataset*) const `[inherited]`

Retrieve the brush to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the brush for.

Returns:

The brush to use for painting.

Definition at line 808 of file KDChartAbstractDiagram.cpp.

```
815 {
```

7.9.2.7 QBrush AbstractDiagram::brush () const `[inherited]`

Retrieve the brush to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The brush to use for painting.

Definition at line 802 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), and KDChart::AbstractDiagram::paintMarker().

```
807 {
```

7.9.2.8 virtual const QPair<QPointF, QPointF> KDChart::AbstractDiagram::calculateDataBoundaries () const `[protected, pure virtual, inherited]`

Implemented in KDChart::BarDiagram, KDChart::LineDiagram, KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

Referenced by KDChart::AbstractDiagram::dataBoundaries().

7.9.2.9 bool AbstractDiagram::checkInvariants (bool *justReturnTheStatus* = false) const `[protected, virtual, inherited]`

Definition at line 930 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane().

Referenced by KDChart::RingDiagram::calculateDataBoundaries(), KDChart::PolarDiagram::calculateDataBoundaries(), KDChart::PieDiagram::calculateDataBoundaries(), KDChart::LineDiagram::calculateDataBoundaries(), KDChart::BarDiagram::calculateDataBoundaries(), KDChart::RingDiagram::paint(), KDChart::PolarDiagram::paint(), KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and KDChart::AbstractDiagram::paintMarker().

```
                               {
        Q_ASSERT_X ( model(), "AbstractDiagram::checkInvariants()",
                    "There is no usable model set, for the diagram." );

        Q_ASSERT_X ( coordinatePlane(), "AbstractDiagram::checkInvariants()",
                    "There is no usable coordinate plane set, for the diagram." );
    }
    return model() && coordinatePlane();
}

int AbstractDiagram::datasetDimension( ) const
```

7.9.2.10 int AbstractPolarDiagram::columnCount () const `[inherited]`

Definition at line 60 of file KDChartAbstractPolarDiagram.cpp.

References KDChart::AbstractPolarDiagram::numberOfValuesPerDataset().

Referenced by KDChart::PieDiagram::calculateDataBoundaries(), KDChart::PieDiagram::paint(), and KDChart::PieDiagram::valueTotals().

```
{
    return static_cast<int>( numberOfValuesPerDataset() );
}
```

7.9.2.11 QModelIndex AbstractDiagram::columnToIndex (int *column*) const `[protected, inherited]`

Definition at line 317 of file KDChartAbstractDiagram.cpp.

```
{
```

7.9.2.12 bool AbstractDiagram::compare (const AbstractDiagram ∗ *other*) const `[inherited]`

Returns true if both diagrams have the same settings.

Definition at line 135 of file KDChartAbstractDiagram.cpp.

```
{
    if( other == this ) return true;
    if( ! other ){
        //qDebug() << "AbstractDiagram::compare() cannot compare to Null pointer";
        return false;
    }
    /*
    qDebug() << "\n             AbstractDiagram::compare() QAbstractScrollArea:";
            // compare QAbstractScrollArea properties
    qDebug() <<
            ((horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
            (verticalScrollBarPolicy()    == other->verticalScrollBarPolicy()));
    qDebug() << "AbstractDiagram::compare() QFrame:";
            // compare QFrame properties
    qDebug() <<
            ((frameShadow() == other->frameShadow()) &&
            (frameShape()   == other->frameShape()) &&
            (frameWidth()   == other->frameWidth()) &&
            (lineWidth()    == other->lineWidth()) &&
            (midLineWidth() == other->midLineWidth()));
    qDebug() << "AbstractDiagram::compare() QAbstractItemView:";
            // compare QAbstractItemView properties
    qDebug() <<
            ((alternatingRowColors() == other->alternatingRowColors()) &&
            (hasAutoScroll()         == other->hasAutoScroll()) &&
#if QT_VERSION > 0x040199
            (dragDropMode()          == other->dragDropMode()) &&
            (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
            (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
            (verticalScrollMode()    == other->verticalScrollMode()) &&
#endif
            (dragEnabled()           == other->dragEnabled()) &&
            (editTriggers()          == other->editTriggers()) &&
            (iconSize()              == other->iconSize()) &&
            (selectionBehavior()     == other->selectionBehavior()) &&
            (selectionMode()         == other->selectionMode()) &&
            (showDropIndicator()     == other->showDropIndicator()) &&
            (tabKeyNavigation()      == other->tabKeyNavigation()) &&
            (textElideMode()         == other->textElideMode()));
    qDebug() << "AbstractDiagram::compare() AttributesModel: ";
            // compare all of the properties stored in the attributes model
    qDebug() << attributesModel()->compare( other->attributesModel() );
    qDebug() << "AbstractDiagram::compare() own:";
            // compare own properties
    qDebug() <<
            ((rootIndex().column()            == other->rootIndex().column()) &&
            (rootIndex().row()                == other->rootIndex().row()) &&
            (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
            (antiAliasing()                   == other->antiAliasing()) &&
            (percentMode()                    == other->percentMode()) &&
            (datasetDimension()               == other->datasetDimension()));
    */
    return  // compare QAbstractScrollArea properties
            (horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
            (verticalScrollBarPolicy()   == other->verticalScrollBarPolicy()) &&
            // compare QFrame properties
            (frameShadow()  == other->frameShadow()) &&
            (frameShape()   == other->frameShape()) &&
            (frameWidth()   == other->frameWidth()) &&
            (lineWidth()    == other->lineWidth()) &&
            (midLineWidth() == other->midLineWidth()) &&
            // compare QAbstractItemView properties
            (alternatingRowColors()  == other->alternatingRowColors()) &&
            (hasAutoScroll()         == other->hasAutoScroll()) &&
#if QT_VERSION > 0x040199
            (dragDropMode()          == other->dragDropMode()) &&
            (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
            (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
            (verticalScrollMode()    == other->verticalScrollMode()) &&
#endif
            (dragEnabled()           == other->dragEnabled()) &&
            (editTriggers()          == other->editTriggers()) &&
            (iconSize()              == other->iconSize()) &&
            (selectionBehavior()     == other->selectionBehavior()) &&
            (selectionMode()         == other->selectionMode()) &&
            (showDropIndicator()     == other->showDropIndicator()) &&
            (tabKeyNavigation()      == other->tabKeyNavigation()) &&
            (textElideMode()         == other->textElideMode()) &&
            // compare all of the properties stored in the attributes model
            attributesModel()->compare( other->attributesModel() ) &&
            // compare own properties
            (rootIndex().column()             == other->rootIndex().column()) &&
            (rootIndex().row()                == other->rootIndex().row()) &&
            (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
            (antiAliasing()                   == other->antiAliasing()) &&
            (percentMode()                    == other->percentMode()) &&
            (datasetDimension()              == other->datasetDimension());
}
```

7.9.2.13 AbstractCoordinatePlane ∗ AbstractDiagram::coordinatePlane () const `[inherited]`

The coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. By default this is a CartesianCoordinatePlane.

Returns:

The coordinate plane associated with the diagram.

Definition at line 226 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractCartesianDiagram::layoutPlanes(), KDChart::PolarDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), KDChart::AbstractPolarDiagram::polarCoordinatePlane(), and KDChart::AbstractCartesianDiagram::setCoordinatePlane().

```
{
    return d->plane;
}
```

7.9.2.14 const QPair< QPointF, QPointF > AbstractDiagram::dataBoundaries () const `[inherited]`

Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

This method returns a chached result of calculations done by calculateDataBoundaries. Classes derived from AbstractDiagram must implement the calculateDataBoundaries function, to specify their own way of calculating the data boundaries. If derived classes want to force recalculation of the data boundaries, they can call setDataBoundariesDirty()

Returned value is in diagram coordinates.

Definition at line 231 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::calculateDataBoundaries(), and d.

Referenced by KDChart::CartesianCoordinatePlane::getRawDataBoundingRectFromDiagrams(), KDChart::PolarCoordinatePlane::layoutDiagrams(), KDChart::LineDiagram::paint(), and KDChart::BarDiagram::paint().

```
{
    if( d->databoundariesDirty ){
        d->databoundaries = calculateDataBoundaries ();
        d->databoundariesDirty = false;
    }
    return d->databoundaries;
}
```

7.9.2.15 void AbstractDiagram::dataChanged (const QModelIndex & *topLeft*, const QModelIndex & *bottomRight*) `[virtual, inherited]`

[reimplemented]

Definition at line 338 of file KDChartAbstractDiagram.cpp.

References d.

```
338 {
339   // We are still too dumb to do intelligent updates...
340   d->databoundariesDirty = true;
341   scheduleDelayedItemsLayout();
342 }
343
344
```

7.9.2.16 void KDChart::AbstractDiagram::dataHidden () `[protected, inherited]`

This signal is emitted, when the hidden status of at least one data cell was (un)set.

7.9.2.17 QList< QBrush > AbstractDiagram::datasetBrushes () const `[inherited]`

The set of dataset brushes currently used, for use in legends, etc.

Note:

Cell-level override brushes, if set, take precedence over the dataset values, so you might need to check these too, in order to find the brush, that is used for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 894 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), KDChart::Legend::datasetCount(), and KDChart::Legend::setBrushesFromDiagram().

```
896                                                                                    {
897         QBrush brush = qVariantValue<QBrush>( attributesModel()->headerData( i, Qt::Vertical, DatasetB
898         ret << brush;
899     }
900
901     return ret;
902 }
903
904 QList<QPen> AbstractDiagram::datasetPens() const
```

7.9.2.18 int AbstractDiagram::datasetDimension () const `[inherited]`

The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

For each dimension it will expect one column of values in the model. If the dimensionality is 1, automatic values will be used for the abscissa.

For example a diagram with the default dimension of 1, will have one column per datapoint (the y values) and will use automatic values for the x axis (1, 2, 3, ... n). If the dimension is 2, the diagram will use the first, (and the third, fifth, etc) columns as X values, and the second, (and the fourth, sixth, etc) column as Y values.

Returns:

The dataset dimension of the diagram.

Definition at line 942 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::calculateDataBoundaries(), KDChart::LineDiagram::getCellValues(), KDChart::CartesianCoordinatePlane::getDataDimensionsList(), KDChart::LineDiagram::paint(), and KDChart::LineDiagram::setType().

```
946 {
```

7.9.2.19 QStringList AbstractDiagram::datasetLabels () const `[inherited]`

The set of dataset labels currently displayed, for use in legends, etc.

Returns:

The set of dataset labels currently displayed.

Definition at line 882 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), and KDChart::Legend::datasetCount().

```
883                                                  : " << attributesModel()->columnCount(attributesModel
884     const int columnCount = attributesModel()->columnCount(attributesModelRootIndex());
885     for( int i = datasetDimension()-1; i < columnCount; i += datasetDimension() ){
886         //qDebug() << "dataset label: " << attributesModel()->headerData( i, Qt::Horizontal, Qt::Displ
887         ret << attributesModel()->headerData( i, Qt::Horizontal, Qt::DisplayRole ).toString();
888     }
889     return ret;
890 }
891
892 QList<QBrush> AbstractDiagram::datasetBrushes() const
```

7.9.2.20 QList< MarkerAttributes > AbstractDiagram::datasetMarkers () const `[inherited]`

The set of dataset markers currently used, for use in legends, etc.

Note:

Cell-level override markers, if set, take precedence over the dataset values, so you might need to check these too, in order to find the marker, that is shown for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 917 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
919                                                                                    {
920         DataValueAttributes a =
921             qVariantValue<DataValueAttributes>( attributesModel()->headerData( i, Qt::Vertical, DataVa
922         const MarkerAttributes &ma = a.markerAttributes();
923         ret << ma;
924     }
925     return ret;
926 }
927
928 bool AbstractDiagram::checkInvariants( bool justReturnTheStatus ) const
```

7.9.2.21 QList< QPen > AbstractDiagram::datasetPens () const `[inherited]`

The set of dataset pens currently used, for use in legends, etc.

Note:

Cell-level override pens, if set, take precedence over the dataset values, so you might need to check these too, in order to find the pens, that is used for a single cell.

Returns:

The current set of dataset pens.

Definition at line 906 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
908                                                                                    {
909         QPen pen = qVariantValue<QPen>( attributesModel()->headerData( i, Qt::Vertical, DatasetPenRole
910         ret << pen;
911     }
912     return ret;
913 }
914
915 QList<MarkerAttributes> AbstractDiagram::datasetMarkers() const
```

7.9.2.22 DataValueAttributes AbstractDiagram::dataValueAttributes (const QModelIndex & *index*) const `[inherited]`

Retrieve the DataValueAttributes for the given index.

This will fall back automatically to what was set at dataset or model level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the attributes for.

Returns:

The DataValueAttributes for the given index.

Definition at line 427 of file KDChartAbstractDiagram.cpp.

```
433 {
```

7.9.2.23 DataValueAttributes AbstractDiagram::dataValueAttributes (int *column*) const `[inherited]`

Retrieve the DataValueAttributes for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the attributes for.

Returns:

The DataValueAttributes for the given dataset.

Definition at line 420 of file KDChartAbstractDiagram.cpp.

```
426 {
```

7.9.2.24 DataValueAttributes AbstractDiagram::dataValueAttributes () const `[inherited]`

Retrieve the DataValueAttributes speficied globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The global DataValueAttributes.

Definition at line 414 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::AbstractDiagram::paintDataValueText(), and KDChart::AbstractDiagram::paintMarker().

```
419 {
```

7.9.2.25 void AbstractDiagram::doItemsLayout () `[virtual, inherited]`

[reimplemented]

Definition at line 329 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::AbstractDiagram::update().

```
329                     {
330         d->plane->layoutDiagrams();
331         update();
332     }
333     QAbstractItemView::doItemsLayout();
334 }
335
336 void AbstractDiagram::dataChanged( const QModelIndex &topLeft,
```

7.9.2.26 qreal AbstractPieDiagram::granularity () const

Returns:

the granularity.

Definition at line 69 of file KDChartAbstractPieDiagram.cpp.

References d.

Referenced by KDChart::PieDiagram::paint().

```
{
    return (d->granularity < 0.05 || d->granularity > 36.0)
            ? 1.0
    : d->granularity;
}
```

7.9.2.27 int AbstractDiagram::horizontalOffset () const `[virtual, inherited]`

[reimplemented]

Definition at line 839 of file KDChartAbstractDiagram.cpp.

```
{ return 0; }
```

7.9.2.28 QModelIndex AbstractDiagram::indexAt (const QPoint & *point*) const `[virtual, inherited]`

[reimplemented]

Definition at line 833 of file KDChartAbstractDiagram.cpp.

```
{ return QModelIndex(); }
```

7.9.2.29 bool AbstractDiagram::isHidden (const QModelIndex & *index*) const `[inherited]`

Retrieve the hidden status for the given index.

This will fall back automatically to what was set at dataset or diagram level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the hidden status for.

Returns:

The hidden status for the given index.

Definition at line 386 of file KDChartAbstractDiagram.cpp.

7.9.2.30 bool AbstractDiagram::isHidden (int *column*) const `[inherited]`

Retrieve the hidden status for the given dataset.

This will fall back automatically to what was set at diagram level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the hidden status for.

Returns:

The hidden status for the given dataset.

Definition at line 379 of file KDChartAbstractDiagram.cpp.

```
385 {
```

7.9.2.31 bool AbstractDiagram::isHidden () const `[inherited]`

Retrieve the hidden status speficied globally.

This will fall back automatically to the default settings (= not hidden), if there are no specific settings.

Returns:

The global hidden status.

Definition at line 373 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), KDChart::LineDiagram::paint(), and KDChart::LineDiagram::valueForCellTesting().

```
378 {
```

7.9.2.32 bool AbstractDiagram::isIndexHidden (const QModelIndex & *index*) const `[virtual, inherited]`

[reimplemented]

Definition at line 845 of file KDChartAbstractDiagram.cpp.

```
847 {}
```

7.9.2.33 QStringList AbstractDiagram::itemRowLabels () const `[inherited]`

The set of item row labels currently displayed, for use in Abscissa axes, etc.

Returns:

The set of item row labels currently displayed.

Definition at line 870 of file KDChartAbstractDiagram.cpp.

```
871                                                  : " << attributesModel()->rowCount(attributesModelRoo
872     const int rowCount = attributesModel()->rowCount(attributesModelRootIndex());
873     for( int i = 0; i < rowCount; ++i ){
874         //qDebug() << "item row label: " << attributesModel()->headerData( i, Qt::Vertical, Qt::Displa
875         ret << attributesModel()->headerData( i, Qt::Vertical, Qt::DisplayRole ).toString();
876     }
877     return ret;
878 }
879
880 QStringList AbstractDiagram::datasetLabels() const
```

7.9.2.34 void KDChart::AbstractDiagram::modelsChanged () `[protected, inherited]`

This signal is emitted, when either the model or the AttributesModel is replaced.

Referenced by KDChart::AbstractDiagram::setAttributesModel(), and KDChart::AbstractDiagram::setModel().

7.9.2.35 QModelIndex AbstractDiagram::moveCursor (CursorAction *cursorAction*, Qt::KeyboardModifiers *modifiers*) `[virtual, inherited]`

[reimplemented]

Definition at line 836 of file KDChartAbstractDiagram.cpp.

```
838 { return 0; }
```

7.9.2.36 virtual double KDChart::AbstractPolarDiagram::numberOfGridRings () const `[pure virtual, inherited]`

Implemented in KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

7.9.2.37 virtual double KDChart::AbstractPolarDiagram::numberOfValuesPerDataset () const `[pure virtual, inherited]`

Implemented in KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

Referenced by KDChart::AbstractPolarDiagram::columnCount().

7.9.2.38 virtual void KDChart::AbstractDiagram::paint (PaintContext ∗ *paintContext*) `[pure virtual, inherited]`

Draw the diagram contents to the rectangle and painter, that are passed in as part of the paint context.

Parameters:

paintContext All information needed for painting.

Implemented in KDChart::BarDiagram, KDChart::LineDiagram, KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

7.9.2.39 void AbstractDiagram::paintDataValueText (QPainter ∗ *painter*, const QModelIndex & *index*, const QPointF & *pos*, double *value*) `[inherited]`

Definition at line 474 of file KDChartAbstractDiagram.cpp.

References KDChart::RelativePosition::alignment(), KDChart::TextAttributes::calculatedFont(), d, KDChart::DataValueAttributes::dataLabel(), KDChart::AbstractDiagram::dataValueAttributes(), KDChart::DataValueAttributes::decimalDigits(), KDChart::TextAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::TextAttributes::pen(), KDChart::DataValueAttributes::position(), KDChart::DataValueAttributes::prefix(), KDChart::TextAttributes::rotation(), KDChart::DataValueAttributes::showRepetitiveDataLabels(), KDChart::DataValueAttributes::suffix(), and KDChart::DataValueAttributes::textAttributes().

Referenced by KDChart::RingDiagram::paint(), and KDChart::PolarDiagram::paint().

```
476 {
477     // paint one data series
478     const DataValueAttributes a( dataValueAttributes(index) );
479     if ( !a.isVisible() ) return;
480 
481     // handle decimal digits
482     int decimalDigits = a.decimalDigits();
483     int decimalPos = QString::number(  value ).indexOf( QLatin1Char( ’.’ ) );
484     QString roundedValue;
485     if ( a.dataLabel().isNull() ) {
486         if ( decimalPos > 0 && value != 0 )
487             roundedValue =  roundValues ( value, decimalPos, decimalDigits );
488         else
489             roundedValue = QString::number(  value );
490     } else
491         roundedValue = a.dataLabel();
492         // handle prefix and suffix
493     if ( !a.prefix().isNull() )
494         roundedValue.prepend( a.prefix() );
495 
496     if ( !a.suffix().isNull() )
497         roundedValue.append( a.suffix() );
498 
499     const TextAttributes ta( a.textAttributes() );
500     // FIXME draw the non-text bits, background, etc
501     if ( ta.isVisible() ) {
502 
503         QPointF pt( pos );
504         /* for debugging:
505         PainterSaver painterSaver( painter );
506         painter->setPen( Qt::black );
507         painter->drawLine( pos - QPointF( 1,1), pos + QPointF( 1,1) );
508         painter->drawLine( pos - QPointF(-1,1), pos + QPointF(-1,1) );
509         */
510 
511         // adjust the text start point position, if alignment is not Bottom/Left
512         const RelativePosition relPos( a.position( value >= 0.0 ) );
513         const Qt::Alignment alignBottomLeft = Qt::AlignBottom | Qt::AlignLeft;
514         const QFont calculatedFont( ta.calculatedFont( d->plane, KDChartEnums::MeasureOrientationMinim
515         //qDebug() << "calculatedFont’s point size:" << calculatedFont.pointSizeF();
516         if( (relPos.alignment() & alignBottomLeft) != alignBottomLeft ){
517             const QRectF boundRect(
518                     d->cachedFontMetrics( calculatedFont, this )->boundingRect( roundedValue ) );
519             if( relPos.alignment() & Qt::AlignRight )
520                 pt.rx() -= boundRect.width();
521             else if( relPos.alignment() & Qt::AlignHCenter )
522                 pt.rx() -= 0.5 * boundRect.width();
523 
524             if( relPos.alignment() & Qt::AlignTop )
525                 pt.ry() += boundRect.height();
```

```
            else if( relPos.alignment() & Qt::AlignVCenter )
                pt.ry() += 0.5 * boundRect.height();
        }

        // FIXME draw the non-text bits, background, etc

        if ( a.showRepetitiveDataLabels() ||
             pos.x() <= d->lastX ||
             d->lastRoundedValue != roundedValue ) {
            d->lastRoundedValue = roundedValue;
            d->lastX = pos.x();

            PainterSaver painterSaver( painter );
            painter->setPen( ta.pen() );
            painter->setFont( calculatedFont );
            painter->translate( pt );
            painter->rotate( ta.rotation() );
            painter->drawText( QPointF(0, 0), roundedValue );
        }
    }
}


```

7.9.2.40 void AbstractDiagram::paintDataValueTexts (QPainter ∗ *painter*) `[protected, virtual, inherited]`

Definition at line 576 of file KDChartAbstractDiagram.cpp.

```
                                                                                {
       for ( int j=0; j< rowCount; ++j ) {
           const QModelIndex index = model()->index( j, i, rootIndex() );
           double value = model()->data( index ).toDouble();
           const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
           paintDataValueText( painter, index, pos, value );
       }
    }
}


```

7.9.2.41 void AbstractDiagram::paintMarker (QPainter ∗ *painter*, const QModelIndex & *index*, const QPointF & *pos*) `[inherited]`

Definition at line 592 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::brush(), KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractDiagram::dataValueAttributes(), KDChart::MarkerAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::DataValueAttributes::markerAttributes(), KDChart::MarkerAttributes::markerColor(), KDChart::MarkerAttributes::markerSize(), KDChart::AbstractDiagram::paintMarker(), and KDChart::MarkerAttributes::pen().

```
{

    if ( !checkInvariants() ) return;
    DataValueAttributes a = dataValueAttributes(index);
    if ( !a.isVisible() ) return;
    const MarkerAttributes &ma = a.markerAttributes();
    if ( !ma.isVisible() ) return;

    PainterSaver painterSaver( painter );
    QSizeF maSize( ma.markerSize() );
    QBrush indexBrush( brush( index ) );
    QPen indexPen( ma.pen() );
    if ( ma.markerColor().isValid() )
        indexBrush.setColor( ma.markerColor() );

    paintMarker( painter, ma, indexBrush, indexPen, pos, maSize );
}


```

7.9.2.42 void AbstractDiagram::paintMarker (QPainter ∗ *painter*, const MarkerAttributes & *markerAttributes*, const QBrush & *brush*, const QPen &, const QPointF & *point*, const QSizeF & *size*) `[virtual, inherited]`

Definition at line 614 of file KDChartAbstractDiagram.cpp.

References KDChart::MarkerAttributes::markerStyle().

Referenced by KDChart::MarkerLayoutItem::paintIntoRect(), and KDChart::AbstractDiagram::paintMarker().

```
{

    const QPen oldPen( painter->pen() );
    // Pen is used to paint 4Pixels - 1 Pixel - Ring and FastCross types.
    // make sure to use the brush color - see above in those cases.
    const bool isFourPixels = (markerAttributes.markerStyle() == MarkerAttributes::Marker4Pixels);
    if( isFourPixels || (markerAttributes.markerStyle() == MarkerAttributes::Marker1Pixel) ){
        // for high-performance point charts with tiny point markers:
        painter->setPen( QPen( brush.color().light() ) );
        if( isFourPixels ){
            const qreal x = pos.x();
            const qreal y = pos.y();
            painter->drawLine( QPointF(x-1.0,y-1.0),
                               QPointF(x+1.0,y-1.0) );
            painter->drawLine( QPointF(x-1.0,y),
                               QPointF(x+1.0,y) );
            painter->drawLine( QPointF(x-1.0,y+1.0),
                               QPointF(x+1.0,y+1.0) );
        }
        painter->drawPoint( pos );
    }else{
        PainterSaver painterSaver( painter );
        // we only a solid line surrounding the markers
        QPen painterPen( pen );
        painterPen.setStyle( Qt::SolidLine );
        painter->setPen( painterPen );
        painter->setBrush( brush );
        painter->setRenderHint ( QPainter::Antialiasing );
        painter->translate( pos );
        switch ( markerAttributes.markerStyle() ) {
            case MarkerAttributes::MarkerCircle:
                painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                            maSize.height(), maSize.width()) );
                break;
            case MarkerAttributes::MarkerSquare:
                {
                    QRectF rect( 0 - maSize.width()/2, 0 - maSize.height()/2,
                                maSize.width(), maSize.height() );
                    painter->drawRect( rect );
                    painter->fillRect( rect, brush.color() );
                    break;
                }
            case MarkerAttributes::MarkerDiamond:
                {
                    QVector <QPointF > diamondPoints;
                    QPointF top, left, bottom, right;
                    top    = QPointF( 0, 0 - maSize.height()/2 );
                    left   = QPointF( 0 - maSize.width()/2, 0 );
                    bottom = QPointF( 0, maSize.height()/2 );
                    right  = QPointF( maSize.width()/2, 0 );
                    diamondPoints << top << left << bottom << right;
                    painter->drawPolygon( diamondPoints );
                    break;
                }
            // both handled on top of the method:
            case MarkerAttributes::Marker1Pixel:
            case MarkerAttributes::Marker4Pixels:
                    break;
            case MarkerAttributes::MarkerRing:
                {
                    painter->setPen( QPen( brush.color() ) );
                    painter->setBrush( Qt::NoBrush );
                    painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                                        maSize.height(), maSize.width()) );
                    break;
                }
            case MarkerAttributes::MarkerCross:
                {
                    QRectF rect( maSize.width()*-0.5, maSize.height()*-0.2,
                                 maSize.width(), maSize.height()*0.4 );
                    painter->drawRect( rect );
                    rect.setTopLeft(QPointF( maSize.width()*-0.2, maSize.height()*-0.5 ));
                    rect.setSize(QSizeF( maSize.width()*0.4, maSize.height() ));
                    painter->drawRect( rect );
                    break;
                }
            case MarkerAttributes::MarkerFastCross:
                {
                    QPointF left, right, top, bottom;
                    left  = QPointF( -maSize.width()/2, 0 );
                    right = QPointF( maSize.width()/2, 0 );
                    top   = QPointF( 0, -maSize.height()/2 );
                    bottom= QPointF( 0, maSize.height()/2 );
                    painter->setPen( QPen( brush.color() ) );
                    painter->drawLine( left, right );
                    painter->drawLine(  top, bottom );
                    break;
                }
            default:
                Q_ASSERT_X ( false, "paintMarkers()",
                            "Type item does not match a defined Marker Type." );
        }
    }
    painter->setPen( oldPen );
}

void AbstractDiagram::paintMarkers( QPainter* painter )
```

7.9.2.43 void AbstractDiagram::paintMarkers (QPainter ∗ *painter*) [protected, virtual, inherited]

Definition at line 716 of file KDChartAbstractDiagram.cpp.

```
                                                                               {
```

```
720         for ( int j=0; j< rowCount; ++j ) {
721             const QModelIndex index = model()->index( j, i, rootIndex() );
722             double value = model()->data( index ).toDouble();
723             const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
724             paintMarker( painter, index, pos );
725         }
726     }
727 }
728
729
```

7.9.2.44 QPen AbstractDiagram::pen (const QModelIndex & *index*) const `[inherited]`

Retrieve the pen to be used, for painting the datapoint at the given index in the model.

Parameters:

index The index of the datapoint in the model.

Returns:

The pen to use for painting.

Definition at line 770 of file KDChartAbstractDiagram.cpp.

```
777 {
```

7.9.2.45 QPen AbstractDiagram::pen (int *dataset*) const `[inherited]`

Retrieve the pen to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the pen for.

Returns:

The pen to use for painting.

Definition at line 762 of file KDChartAbstractDiagram.cpp.

```
769 {
```

7.9.2.46 QPen AbstractDiagram::pen () const `[inherited]`

Retrieve the pen to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The pen to use for painting.

Definition at line 756 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::PieDiagram::paint(), and KDChart::LineDiagram::paint().

```
761 {
```

7.9.2.47 bool AbstractDiagram::percentMode () const `[inherited]`

Definition at line 468 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::CartesianCoordinatePlane::getDataDimensionsList().

7.9.2.48 PieAttributes AbstractPieDiagram::pieAttributes (const QModelIndex & *index*) const

Definition at line 121 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::PieAttributesRole.

```
{
    return qVariantValue<PieAttributes>(
        d->attributesModel->data(
            d->attributesModel->mapFromSource( index ),
            PieAttributesRole ) );
}
```

7.9.2.49 PieAttributes AbstractPieDiagram::pieAttributes (int *column*) const

Definition at line 113 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::PieAttributesRole.

```
{
    return qVariantValue<PieAttributes>(
        d->attributesModel->data(
            d->attributesModel->mapFromSource( columnToIndex( column ) ).column(),
            PieAttributesRole ) );
}
```

7.9.2.50 PieAttributes AbstractPieDiagram::pieAttributes () const

Definition at line 104 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::PieAttributesRole.

Referenced by KDChart::PieDiagram::calculateDataBoundaries(), and KDChart::PieDiagram::paint().

```
{
    return qVariantValue<PieAttributes>(
        d->attributesModel->data( PieAttributesRole ) );
}
```

7.9.2.51 const PolarCoordinatePlane ∗ AbstractPolarDiagram::polarCoordinatePlane () const `[inherited]`

Definition at line 55 of file KDChartAbstractPolarDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane().

Referenced by KDChart::PieDiagram::paint().

```
56 {
57     return dynamic_cast<const PolarCoordinatePlane*>( coordinatePlane() );
58 }
```

7.9.2.52 void KDChart::AbstractDiagram::propertiesChanged () `[protected, inherited]`

Emitted upon change of a property of the Diagram.

Referenced by KDChart::LineDiagram::resetLineAttributes(), KDChart::AbstractDiagram::setDataValueAttributes(), KDChart::LineDiagram::setLineAttributes(), KDChart::LineDiagram::setThreeDLineAttributes(), and KDChart::LineDiagram::setType().

7.9.2.53 virtual void KDChart::AbstractDiagram::resize (const QSizeF & *area*) `[pure virtual, inherited]`

Called by the widget's sizeEvent.

Adjust all internal structures, that are calculated, dependending on the size of the widget.

Parameters:

area

Implemented in KDChart::BarDiagram, KDChart::LineDiagram, KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

7.9.2.54 void AbstractDiagram::scrollTo (const QModelIndex & *index*, ScrollHint *hint* = EnsureVisible) `[virtual, inherited]`

[reimplemented]

Definition at line 830 of file KDChartAbstractDiagram.cpp.

```
832 { return QModelIndex(); }
```

7.9.2.55 void AbstractDiagram::setAllowOverlappingDataValueTexts (bool *allow*) `[inherited]`

Set whether data value labels are allowed to overlap.

Parameters:

allow True means that overlapping labels are allowed.

Definition at line 440 of file KDChartAbstractDiagram.cpp.

References d.

```
445 {
```

7.9.2.56 void AbstractDiagram::setAntiAliasing (bool *enabled*) `[inherited]`

Set whether anti-aliasing is to be used while rendering this diagram.

Parameters:

enabled True means that AA is enabled.

Definition at line 451 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.9.2.57 void AbstractDiagram::setAttributesModel (AttributesModel ∗ *model*) `[virtual, inherited]`

Associate an AttributesModel with this diagram.

Note that the diagram does _not_ take ownership of the AttributesModel. This should thus only be used with AttributesModels that have been explicitely created by the user, and are owned by her. Setting an AttributesModel that is internal to another diagram is an error.

Correct:

```
AttributesModel *am = new AttributesModel( model, 0 );
diagram1->setAttributesModel( am );
diagram2->setAttributesModel( am );
```

Wrong:

```
diagram1->setAttributesModel( diagram2->attributesModel() );
```

Parameters:

model The AttributesModel to use for this diagram.

See also:

AttributesModel, usesExternalAttributesModel

Definition at line 261 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::AbstractDiagram::modelsChanged().

```
{
    if( amodel->sourceModel() != model() ) {
        qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
                 "Trying to set an attributesmodel which works on a different "
                 "model than the diagram.");
        return;
    }
    if( qobject_cast<PrivateAttributesModel*>(amodel) ) {
        qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
                 "Trying to set an attributesmodel that is private to another diagram.");
        return;
    }
    d->setAttributesModel(amodel);
    scheduleDelayedItemsLayout();
    d->databoundariesDirty = true;
    emit modelsChanged();
}
```

7.9.2.58 void AbstractDiagram::setAttributesModelRootIndex (const QModelIndex & *idx*) `[protected, inherited]`

Definition at line 301 of file KDChartAbstractDiagram.cpp.

References d.

7.9.2.59 void AbstractDiagram::setBrush (const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting all datasets in the model.

Parameters:

brush The brush to use.

Definition at line 786 of file KDChartAbstractDiagram.cpp.

```
792 {
```

7.9.2.60 void AbstractDiagram::setBrush (int *dataset*, const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting the given dataset.

Parameters:

dataset The dataset's column in the model.

pen The brush to use.

Definition at line 793 of file KDChartAbstractDiagram.cpp.

```
801 {
```

7.9.2.61 void AbstractDiagram::setBrush (const QModelIndex & *index*, const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting the datapoint at the given index.

Parameters:

index The datapoint's index in the model.

brush The brush to use.

Definition at line 778 of file KDChartAbstractDiagram.cpp.

```
785 {
```

7.9.2.62 void AbstractDiagram::setCoordinatePlane (AbstractCoordinatePlane ∗ *plane*) `[virtual, inherited]`

Set the coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. The chart takes ownership.

Returns:

The coordinate plane associated with the diagram.

Reimplemented in KDChart::AbstractCartesianDiagram.

Definition at line 324 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractCoordinatePlane::addDiagram(), KDChart::AbstractCartesianDiagram::setCoordinatePlane(), and KDChart::AbstractCoordinatePlane::takeDiagram().

```
328 {
```

7.9.2.63 void AbstractDiagram::setDataBoundariesDirty () const `[protected, inherited]`

Definition at line 240 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::BarDiagram::setThreeDBarAttributes(), KDChart::LineDiagram::setThreeDLineAttributes(), KDChart::LineDiagram::setType(), and KDChart::BarDiagram::setType().

```
241 {
242     d->databoundariesDirty = true;
243 }
```

7.9.2.64 void AbstractDiagram::setDatasetDimension (int *dimension*) `[inherited]`

Sets the dataset dimension of the diagram.

See also:

datasetDimension.

Parameters:

dimension

Definition at line 947 of file KDChartAbstractDiagram.cpp.

References d.

```
954 {
```

7.9.2.65 void AbstractDiagram::setDataValueAttributes (const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for all datapoints in the model.

Parameters:

a The attributes to set.

Definition at line 434 of file KDChartAbstractDiagram.cpp.

References d.

```
439 {
```

7.9.2.66 void AbstractDiagram::setDataValueAttributes (int *dataset*, const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for the given dataset.

Parameters:

dataset The dataset to set the attributes for.

a The attributes to set.

Definition at line 406 of file KDChartAbstractDiagram.cpp.

References d.

```
413 {
```

7.9.2.67 void AbstractDiagram::setDataValueAttributes (const QModelIndex & *index*, const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for the given index.

Parameters:

index The datapoint to set the attributes for.

a The attributes to set.

Definition at line 395 of file KDChartAbstractDiagram.cpp.

References d, KDChart::DataValueLabelAttributesRole, and KDChart::AbstractDiagram::propertiesChanged().

```
395 {
396     d->attributesModel->setData(
397         d->attributesModel->mapFromSource( index ),
398         qVariantFromValue( a ),
399         DataValueLabelAttributesRole );
400     emit propertiesChanged();
401 }
402
403
```

7.9.2.68 void AbstractPieDiagram::setGranularity (qreal *value*)

Set the granularity: the smaller the granularity the more your diagram segments will show facettes instead of rounded segments.

Parameters:

value the granularity value between 0.05 (one twentieth of a degree) and 36.0 (one tenth of a full circle), other values will be interpreted as 1.0.

Definition at line 64 of file KDChartAbstractPieDiagram.cpp.

References d.

```
{
    d->granularity = value;
}
```

7.9.2.69 void AbstractDiagram::setHidden (bool *hidden*) `[inherited]`

Hide (or unhide, resp.) all datapoints in the model.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

hidden The hidden status to set.

Definition at line 365 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.9.2.70 void AbstractDiagram::setHidden (int *column*, bool *hidden*) `[inherited]`

Hide (or unhide, resp.) a dataset.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

dataset The dataset to set the hidden status for.

hidden The hidden status to set.

Definition at line 356 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.9.2.71 void AbstractDiagram::setHidden (const QModelIndex & *index*, bool *hidden*) `[inherited]`

Hide (or unhide, resp.) a data cell.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

index The datapoint to set the hidden status for.

hidden The hidden status to set.

Definition at line 347 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::DataHiddenRole.

```
355 {
```

7.9.2.72 void AbstractDiagram::setModel (QAbstractItemModel ∗ *model*) `[virtual, inherited]`

Associate a model with the diagram.

Definition at line 245 of file KDChartAbstractDiagram.cpp.

References d, KDChart::AttributesModel::initFrom(), and KDChart::AbstractDiagram::modelsChanged().

```
246 {
247   QAbstractItemView::setModel( newModel );
248   AttributesModel* amodel = new PrivateAttributesModel( newModel, this );
249   amodel->initFrom( d->attributesModel );
250   d->setAttributesModel(amodel);
251   scheduleDelayedItemsLayout();
252   d->databoundariesDirty = true;
253   emit modelsChanged();
254 }
```

7.9.2.73 void AbstractDiagram::setPen (const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting all datasets in the model.

Parameters:

pen The pen to use.

Definition at line 740 of file KDChartAbstractDiagram.cpp.

```
746 {
```

7.9.2.74 void AbstractDiagram::setPen (int *dataset*, const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting the given dataset.

Parameters:

- ***dataset*** The dataset's row in the model.
- ***pen*** The pen to use.

Definition at line 747 of file KDChartAbstractDiagram.cpp.

```
{
```

7.9.2.75 void AbstractDiagram::setPen (const QModelIndex & *index*, const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting the datapoint at the given index.

Parameters:

- ***index*** The datapoint's index in the model.
- ***pen*** The pen to use.

Definition at line 732 of file KDChartAbstractDiagram.cpp.

```
{
```

7.9.2.76 void AbstractDiagram::setPercentMode (bool *percent*) `[inherited]`

Definition at line 462 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::setType(), and KDChart::BarDiagram::setType().

```
{
```

7.9.2.77 void AbstractPieDiagram::setPieAttributes (int *column*, const PieAttributes & *a*)

Definition at line 94 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::PieAttributesRole.

```
{
    d->attributesModel->setHeaderData(
        column, Qt::Vertical, qVariantFromValue( attrs ), PieAttributesRole );
    emit layoutChanged( this );
}
```

7.9.2.78 void AbstractPieDiagram::setPieAttributes (const PieAttributes & *a*)

Definition at line 88 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::PieAttributesRole.

```
89 {
90     d->attributesModel->setModelData( qVariantFromValue( attrs ), PieAttributesRole );
91     emit layoutChanged( this );
92 }
```

7.9.2.79 void AbstractDiagram::setRootIndex (const QModelIndex & *idx*) `[virtual, inherited]`

Set the root index in the model, where the diagram starts referencing data for display.

[reimplemented]

Definition at line 294 of file KDChartAbstractDiagram.cpp.

References d.

7.9.2.80 void AbstractDiagram::setSelection (const QRect & *rect*, QItemSelectionModel::SelectionFlags *command*) `[virtual, inherited]`

[reimplemented]

Definition at line 848 of file KDChartAbstractDiagram.cpp.

```
850 { return QRegion(); }
```

7.9.2.81 void AbstractPieDiagram::setStartPosition (int *degrees*)

Deprecated

Use PolarCoordinatePlane::setStartPosition(qreal degrees) instead.

Definition at line 77 of file KDChartAbstractPieDiagram.cpp.

```
78 {
79     qWarning() << "Deprecated AbstractPieDiagram::setStartPosition() called, setting ignored.";
80 }
```

7.9.2.82 void AbstractPieDiagram::setThreeDPieAttributes (const QModelIndex & *index*, const ThreeDPieAttributes & *a*)

Definition at line 143 of file KDChartAbstractPieDiagram.cpp.

References KDChart::ThreeDPieAttributesRole.

```
144 {
145     model()->setData( index, qVariantFromValue( tda ), ThreeDPieAttributesRole );
146     emit layoutChanged( this );
147 }
```

7.9.2.83 void AbstractPieDiagram::setThreeDPieAttributes (int *column*, const ThreeDPieAttributes & *a*)

Definition at line 136 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::ThreeDPieAttributesRole.

```
{
    d->attributesModel->setHeaderData(
        column, Qt::Vertical, qVariantFromValue( tda ), ThreeDPieAttributesRole );
    emit layoutChanged( this );
}
```

7.9.2.84 void AbstractPieDiagram::setThreeDPieAttributes (const ThreeDPieAttributes & *a*)

Definition at line 130 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::ThreeDPieAttributesRole.

```
{
    d->attributesModel->setModelData( qVariantFromValue( tda ), ThreeDPieAttributesRole );
    emit layoutChanged( this );
}
```

7.9.2.85 int AbstractPieDiagram::startPosition () const

Deprecated

Use qreal PolarCoordinatePlane::startPosition instead.

Definition at line 82 of file KDChartAbstractPieDiagram.cpp.

```
{
    qWarning() << "Deprecated AbstractPieDiagram::startPosition() called.";
    return 0;
}
```

7.9.2.86 ThreeDPieAttributes AbstractPieDiagram::threeDPieAttributes (const QModelIndex & *index*) const

Definition at line 169 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::ThreeDPieAttributesRole.

```
{
    return qVariantValue<ThreeDPieAttributes>(
        d->attributesModel->data(
            d->attributesModel->mapFromSource( index ),
            ThreeDPieAttributesRole ) );
}
```

7.9.2.87 ThreeDPieAttributes AbstractPieDiagram::threeDPieAttributes (int *column*) const

Definition at line 161 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::ThreeDPieAttributesRole.

```
162 {
163     return qVariantValue<ThreeDPieAttributes>(
164         d->attributesModel->data(
165             d->attributesModel->mapFromSource( columnToIndex( column ) ).column(),
166             ThreeDPieAttributesRole ) );
167 }
```

7.9.2.88 ThreeDPieAttributes AbstractPieDiagram::threeDPieAttributes () const

Definition at line 152 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::ThreeDPieAttributesRole.

Referenced by KDChart::PieDiagram::paint().

```
153 {
154     return qVariantValue<ThreeDPieAttributes>(
155         d->attributesModel->data( ThreeDPieAttributesRole ) );
156 }
```

7.9.2.89 void AbstractDiagram::update () const `[inherited]`

Definition at line 961 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractDiagram::doItemsLayout().

7.9.2.90 void KDChart::AbstractDiagram::useDefaultColors () `[inherited]`

Set the palette to be used, for painting datasets to the default palette.

See also:

KDChart::Palette. FIXME: fold into one usePalette (KDChart::Palette&) method

Definition at line 855 of file KDChartAbstractDiagram.cpp.

References d.

```
859 {
```

7.9.2.91 void KDChart::AbstractDiagram::useRainbowColors () `[inherited]`

Set the palette to be used, for painting datasets to the rainbow palette.

See also:

KDChart::Palette.

Definition at line 865 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.9.2.92 bool AbstractDiagram::usesExternalAttributesModel () const `[virtual, inherited]`

Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

See also:

setAttributesModel

Definition at line 280 of file KDChartAbstractDiagram.cpp.

References d.

```
{
    return d->usesExternalAttributesModel();
}
```

7.9.2.93 void KDChart::AbstractDiagram::useSubduedColors () `[inherited]`

Set the palette to be used, for painting datasets to the subdued palette.

See also:

KDChart::Palette.

Definition at line 860 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.9.2.94 double AbstractDiagram::valueForCell (int *row*, int *column*) const `[protected, inherited]`

Helper method, retrieving the data value (DisplayRole) for a given row and column.

Parameters:

row The row to query.

column The column to query.

Returns:

The value of the display role at the given row and column as a double.

Definition at line 955 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::attributesModelRootIndex(), and d.

Referenced by KDChart::LineDiagram::paint().

```
{
```

7.9.2.95 virtual double KDChart::AbstractPolarDiagram::valueTotals () const `[pure virtual, inherited]`

Implemented in KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

Referenced by KDChart::PolarCoordinatePlane::layoutDiagrams().

7.9.2.96 int AbstractDiagram::verticalOffset () const `[virtual, inherited]`

[reimplemented]

Definition at line 842 of file KDChartAbstractDiagram.cpp.

```
844 { return true; }
```

7.9.2.97 QRect AbstractDiagram::visualRect (const QModelIndex & *index*) const `[virtual, inherited]`

[reimplemented]

Definition at line 825 of file KDChartAbstractDiagram.cpp.

```
829 {}
```

7.9.2.98 QRegion AbstractDiagram::visualRegionForSelection (const QItemSelection & *selection*) const `[virtual, inherited]`

[reimplemented]

Definition at line 851 of file KDChartAbstractDiagram.cpp.

7.9.3 Member Data Documentation

7.9.3.1 Q_SIGNALS KDChart::AbstractDiagram::__pad0__ `[protected, inherited]`

Definition at line 589 of file KDChartAbstractDiagram.h.

The documentation for this class was generated from the following files:

- KDChartAbstractPieDiagram.h
- KDChartAbstractPieDiagram.cpp

7.10 KDChart::AbstractPolarDiagram Class Reference

`#include <KDChartAbstractPolarDiagram.h>`

Inheritance diagram for KDChart::AbstractPolarDiagram:Collaboration diagram for KDChart::AbstractPolarDiagram:

Public Member Functions

- AbstractPolarDiagram (QWidget ∗parent=0, PolarCoordinatePlane ∗plane=0)
- bool allowOverlappingDataValueTexts () const
- bool antiAliasing () const
- virtual AttributesModel ∗ attributesModel () const

  Returns the AttributesModel, that is used by this diagram.

- QBrush brush (const QModelIndex &index) const

  Retrieve the brush to be used, for painting the datapoint at the given index in the model.

- QBrush brush (int dataset) const

  Retrieve the brush to be used for the given dataset.

- QBrush brush () const

  Retrieve the brush to be used for painting datapoints globally.

- int columnCount () const
- bool compare (const AbstractDiagram ∗other) const

  Returns true if both diagrams have the same settings.

- AbstractCoordinatePlane ∗ coordinatePlane () const

  The coordinate plane associated with the diagram.

- const QPair< QPointF, QPointF > dataBoundaries () const

  Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

- virtual void dataChanged (const QModelIndex &topLeft, const QModelIndex &bottomRight)

  [reimplemented]

- QList< QBrush > datasetBrushes () const

  The set of dataset brushes currently used, for use in legends, etc.

- int datasetDimension () const

  The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

- QStringList datasetLabels () const

  The set of dataset labels currently displayed, for use in legends, etc.

- QList< MarkerAttributes > datasetMarkers () const

  The set of dataset markers currently used, for use in legends, etc.

- QList< QPen > datasetPens () const

  The set of dataset pens currently used, for use in legends, etc.

- DataValueAttributes dataValueAttributes (const QModelIndex &index) const

  Retrieve the DataValueAttributes for the given index.

- DataValueAttributes dataValueAttributes (int column) const

  Retrieve the DataValueAttributes for the given dataset.

- DataValueAttributes dataValueAttributes () const

  Retrieve the DataValueAttributes speficied globally.

- virtual void doItemsLayout ()

  [reimplemented]

- virtual int horizontalOffset () const

  [reimplemented]

- virtual QModelIndex indexAt (const QPoint &point) const

  [reimplemented]

- bool isHidden (const QModelIndex &index) const

  Retrieve the hidden status for the given index.

- bool isHidden (int column) const

  Retrieve the hidden status for the given dataset.

- bool isHidden () const

  Retrieve the hidden status speficied globally.

- virtual bool isIndexHidden (const QModelIndex &index) const

  [reimplemented]

- QStringList itemRowLabels () const

  The set of item row labels currently displayed, for use in Abscissa axes, etc.

- virtual QModelIndex moveCursor (CursorAction cursorAction, Qt::KeyboardModifiers modifiers)

  [reimplemented]

- virtual double numberOfGridRings () const=0
- virtual double numberOfValuesPerDataset () const=0
- virtual void paint (PaintContext ∗paintContext)=0

  Draw the diagram contents to the rectangle and painter, that are passed in as part of the paint context.

- void paintDataValueText (QPainter ∗painter, const QModelIndex &index, const QPointF &pos, double value)
- void paintMarker (QPainter ∗painter, const QModelIndex &index, const QPointF &pos)
- virtual void paintMarker (QPainter ∗painter, const MarkerAttributes &markerAttributes, const QBrush &brush, const QPen &, const QPointF &point, const QSizeF &size)

- QPen pen (const QModelIndex &index) const

  Retrieve the pen to be used, for painting the datapoint at the given index in the model.

- QPen pen (int dataset) const

  Retrieve the pen to be used for the given dataset.

- QPen pen () const

  Retrieve the pen to be used for painting datapoints globally.

- bool percentMode () const
- const PolarCoordinatePlane ∗ polarCoordinatePlane () const
- virtual void resize (const QSizeF &area)=0

  Called by the widget's sizeEvent.

- virtual void scrollTo (const QModelIndex &index, ScrollHint hint=EnsureVisible)

  [reimplemented]

- void setAllowOverlappingDataValueTexts (bool allow)

  Set whether data value labels are allowed to overlap.

- void setAntiAliasing (bool enabled)

  Set whether anti-aliasing is to be used while rendering this diagram.

- virtual void setAttributesModel (AttributesModel ∗model)

  Associate an AttributesModel with this diagram.

- void setBrush (const QBrush &brush)

  Set the brush to be used, for painting all datasets in the model.

- void setBrush (int dataset, const QBrush &brush)

  Set the brush to be used, for painting the given dataset.

- void setBrush (const QModelIndex &index, const QBrush &brush)

  Set the brush to be used, for painting the datapoint at the given index.

- virtual void setCoordinatePlane (AbstractCoordinatePlane ∗plane)

  Set the coordinate plane associated with the diagram.

- void setDatasetDimension (int dimension)

  Sets the dataset dimension of the diagram.

- void setDataValueAttributes (const DataValueAttributes &a)

  Set the DataValueAttributes for all datapoints in the model.

- void setDataValueAttributes (int dataset, const DataValueAttributes &a)

  Set the DataValueAttributes for the given dataset.

- void setDataValueAttributes (const QModelIndex &index, const DataValueAttributes &a)

  Set the DataValueAttributes for the given index.

- void setHidden (bool hidden)

  Hide (or unhide, resp.) all datapoints in the model.

- void setHidden (int column, bool hidden)

  Hide (or unhide, resp.) a dataset.

- void setHidden (const QModelIndex &index, bool hidden)

  Hide (or unhide, resp.) a data cell.

- virtual void setModel (QAbstractItemModel ∗model)

  Associate a model with the diagram.

- void setPen (const QPen &pen)

  Set the pen to be used, for painting all datasets in the model.

- void setPen (int dataset, const QPen &pen)

  Set the pen to be used, for painting the given dataset.

- void setPen (const QModelIndex &index, const QPen &pen)

  Set the pen to be used, for painting the datapoint at the given index.

- void setPercentMode (bool percent)
- virtual void setRootIndex (const QModelIndex &idx)

  Set the root index in the model, where the diagram starts referencing data for display.

- virtual void setSelection (const QRect &rect, QItemSelectionModel::SelectionFlags command)

  [reimplemented]

- void update () const
- void useDefaultColors ()

  Set the palette to be used, for painting datasets to the default palette.

- void useRainbowColors ()

  Set the palette to be used, for painting datasets to the rainbow palette.

- virtual bool usesExternalAttributesModel () const

  Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

- void useSubduedColors ()

  Set the palette to be used, for painting datasets to the subdued palette.

- virtual double valueTotals () const=0
- virtual int verticalOffset () const

  [reimplemented]

- virtual QRect visualRect (const QModelIndex &index) const

  [reimplemented]

- virtual QRegion visualRegionForSelection (const QItemSelection &selection) const

[reimplemented]

- virtual ∼AbstractPolarDiagram ()

Protected Member Functions

- QModelIndex attributesModelRootIndex () const
- virtual const QPair< QPointF, QPointF > calculateDataBoundaries () const=0
- virtual bool checkInvariants (bool justReturnTheStatus=false) const
- QModelIndex columnToIndex (int column) const
- void dataHidden ()

  This signal is emitted, when the hidden status of at least one data cell was (un)set.

- void modelsChanged ()

  This signal is emitted, when either the model or the AttributesModel is replaced.

- virtual void paintDataValueTexts (QPainter ∗painter)
- virtual void paintMarkers (QPainter ∗painter)
- void propertiesChanged ()

  Emitted upon change of a property of the Diagram.

- void setAttributesModelRootIndex (const QModelIndex &)
- void setDataBoundariesDirty () const
- double valueForCell (int row, int column) const

  Helper method, retrieving the data value (DisplayRole) for a given row and column.

Protected Attributes

- Q_SIGNALS __pad0__: void layoutChanged(AbstractDiagram∗)

7.10.1 Constructor & Destructor Documentation

7.10.1.1 AbstractPolarDiagram::AbstractPolarDiagram (QWidget ∗ *parent* = 0, PolarCoordinatePlane ∗ *plane* = 0) `[explicit]`

Definition at line 48 of file KDChartAbstractPolarDiagram.cpp.

```
50     : AbstractDiagram ( new Private(), parent, plane )
51 {
52 }
```

7.10.1.2 virtual KDChart::AbstractPolarDiagram::∼AbstractPolarDiagram () `[virtual]`

Definition at line 45 of file KDChartAbstractPolarDiagram.h.

```
45 {}
```

7.10.2 Member Function Documentation

7.10.2.1 bool AbstractDiagram::allowOverlappingDataValueTexts () const `[inherited]`

Returns:

Whether data value labels are allowed to overlap.

Definition at line 446 of file KDChartAbstractDiagram.cpp.

References d.

```
450 {
```

7.10.2.2 bool AbstractDiagram::antiAliasing () const `[inherited]`

Returns:

Whether anti-aliasing is to be used for rendering this diagram.

Definition at line 457 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::paint().

```
461 {
```

7.10.2.3 AttributesModel ∗ AbstractDiagram::attributesModel () const `[virtual, inherited]`

Returns the AttributesModel, that is used by this diagram.

By default each diagram owns its own AttributesModel, which should never be deleted. Only if a user-supplied AttributesModel has been set does the pointer returned here not belong to the diagram.

Returns:

The AttributesModel associated with the diagram.

See also:

setAttributesModel

Definition at line 286 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::RingDiagram::paint(), KDChart::PolarDiagram::paint(), and KDChart::BarDiagram::setBarAttributes().

```
287 {
288     return d->attributesModel;
289 }
```

7.10.2.4 QModelIndex AbstractDiagram::attributesModelRootIndex () const `[protected, inherited]`

returns a QModelIndex pointing into the AttributesModel that corresponds to the root index of the diagram.

Definition at line 310 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::calculateDataBoundaries(), KDChart::BarDiagram::calculateDataBoundaries(), KDChart::LineDiagram::numberOfAbscissaSegments(), KDChart::BarDiagram::numberOfAbscissaSegments(), KDChart::LineDiagram::numberOfOrdinateSegments(), KDChart::BarDiagram::numberOfOrdinateSegments(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and KDChart::AbstractDiagram::valueForCell().

```
316 {
```

7.10.2.5 QBrush AbstractDiagram::brush (const QModelIndex & *index*) const `[inherited]`

Retrieve the brush to be used, for painting the datapoint at the given index in the model.

Parameters:

index The index of the datapoint in the model.

Returns:

The brush to use for painting.

Definition at line 816 of file KDChartAbstractDiagram.cpp.

```
822                                 :
QRect AbstractDiagram::visualRect(const QModelIndex &) const
```

7.10.2.6 QBrush AbstractDiagram::brush (int *dataset*) const `[inherited]`

Retrieve the brush to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the brush for.

Returns:

The brush to use for painting.

Definition at line 808 of file KDChartAbstractDiagram.cpp.

```
815 {
```

7.10.2.7 QBrush AbstractDiagram::brush () const `[inherited]`

Retrieve the brush to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The brush to use for painting.

Definition at line 802 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), and KDChart::AbstractDiagram::paintMarker().

```
807 {
```

7.10.2.8 virtual const QPair<QPointF, QPointF> KDChart::AbstractDiagram::calculateDataBoundaries () const `[protected, pure virtual, inherited]`

Implemented in KDChart::BarDiagram, KDChart::LineDiagram, KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

Referenced by KDChart::AbstractDiagram::dataBoundaries().

7.10.2.9 bool AbstractDiagram::checkInvariants (bool *justReturnTheStatus* = false) const `[protected, virtual, inherited]`

Definition at line 930 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane().

Referenced by KDChart::RingDiagram::calculateDataBoundaries(), KDChart::PolarDiagram::calculateDataBoundaries(), KDChart::PieDiagram::calculateDataBoundaries(), KDChart::LineDiagram::calculateDataBoundaries(), KDChart::BarDiagram::calculateDataBoundaries(), KDChart::RingDiagram::paint(), KDChart::PolarDiagram::paint(), KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and KDChart::AbstractDiagram::paintMarker().

```
930                                 {
931         Q_ASSERT_X ( model(), "AbstractDiagram::checkInvariants()",
932                     "There is no usable model set, for the diagram." );
933
934         Q_ASSERT_X ( coordinatePlane(), "AbstractDiagram::checkInvariants()",
935                     "There is no usable coordinate plane set, for the diagram." );
936     }
937     return model() && coordinatePlane();
938 }
939
940 int AbstractDiagram::datasetDimension( ) const
```

7.10.2.10 int AbstractPolarDiagram::columnCount () const

Definition at line 60 of file KDChartAbstractPolarDiagram.cpp.

References numberOfValuesPerDataset().

Referenced by KDChart::PieDiagram::calculateDataBoundaries(), KDChart::PieDiagram::paint(), and KDChart::PieDiagram::valueTotals().

```
61 {
62     return static_cast<int>( numberOfValuesPerDataset() );
63 }
```

7.10.2.11 QModelIndex AbstractDiagram::columnToIndex (int *column*) const `[protected, inherited]`

Definition at line 317 of file KDChartAbstractDiagram.cpp.

```
323 {
```

7.10.2.12 bool AbstractDiagram::compare (const AbstractDiagram ∗ *other*) const `[inherited]`

Returns true if both diagrams have the same settings.

Definition at line 135 of file KDChartAbstractDiagram.cpp.

```
136 {
137     if( other == this ) return true;
138     if( ! other ){
139         //qDebug() << "AbstractDiagram::compare() cannot compare to Null pointer";
140         return false;
141     }
142     /*
143     qDebug() << "\n             AbstractDiagram::compare() QAbstractScrollArea:";
144             // compare QAbstractScrollArea properties
145     qDebug() <<
146             ((horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
147             (verticalScrollBarPolicy()    == other->verticalScrollBarPolicy()));
148     qDebug() << "AbstractDiagram::compare() QFrame:";
149             // compare QFrame properties
150     qDebug() <<
151             ((frameShadow() == other->frameShadow()) &&
152             (frameShape()   == other->frameShape()) &&
153             (frameWidth()   == other->frameWidth()) &&
154             (lineWidth()    == other->lineWidth()) &&
155             (midLineWidth() == other->midLineWidth()));
156     qDebug() << "AbstractDiagram::compare() QAbstractItemView:";
157             // compare QAbstractItemView properties
158     qDebug() <<
159             ((alternatingRowColors() == other->alternatingRowColors()) &&
160             (hasAutoScroll()         == other->hasAutoScroll()) &&
161 #if QT_VERSION > 0x040199
162             (dragDropMode()          == other->dragDropMode()) &&
163             (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
164             (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
165             (verticalScrollMode()    == other->verticalScrollMode()) &&
166 #endif
167             (dragEnabled()           == other->dragEnabled()) &&
168             (editTriggers()          == other->editTriggers()) &&
169             (iconSize()              == other->iconSize()) &&
170             (selectionBehavior()     == other->selectionBehavior()) &&
171             (selectionMode()         == other->selectionMode()) &&
172             (showDropIndicator()     == other->showDropIndicator()) &&
173             (tabKeyNavigation()      == other->tabKeyNavigation()) &&
            (textElideMode()         == other->textElideMode()));
    qDebug() << "AbstractDiagram::compare() AttributesModel: ";
            // compare all of the properties stored in the attributes model
    qDebug() << attributesModel()->compare( other->attributesModel() );
    qDebug() << "AbstractDiagram::compare() own:";
            // compare own properties
    qDebug() <<
            ((rootIndex().column()            == other->rootIndex().column()) &&
            (rootIndex().row()                == other->rootIndex().row()) &&
            (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
            (antiAliasing()                   == other->antiAliasing()) &&
            (percentMode()                    == other->percentMode()) &&
            (datasetDimension()               == other->datasetDimension()));
    */
    return  // compare QAbstractScrollArea properties
            (horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
            (verticalScrollBarPolicy()   == other->verticalScrollBarPolicy()) &&
            // compare QFrame properties
            (frameShadow()  == other->frameShadow()) &&
            (frameShape()   == other->frameShape()) &&
            (frameWidth()   == other->frameWidth()) &&
            (lineWidth()    == other->lineWidth()) &&
            (midLineWidth() == other->midLineWidth()) &&
            // compare QAbstractItemView properties
            (alternatingRowColors()  == other->alternatingRowColors()) &&
            (hasAutoScroll()         == other->hasAutoScroll()) &&
#if QT_VERSION > 0x040199
            (dragDropMode()          == other->dragDropMode()) &&
            (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
            (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
            (verticalScrollMode()    == other->verticalScrollMode()) &&
#endif
            (dragEnabled()           == other->dragEnabled()) &&
            (editTriggers()          == other->editTriggers()) &&
            (iconSize()              == other->iconSize()) &&
            (selectionBehavior()     == other->selectionBehavior()) &&
            (selectionMode()         == other->selectionMode()) &&
            (showDropIndicator()     == other->showDropIndicator()) &&
            (tabKeyNavigation()      == other->tabKeyNavigation()) &&
            (textElideMode()         == other->textElideMode()) &&
            // compare all of the properties stored in the attributes model
            attributesModel()->compare( other->attributesModel() ) &&
            // compare own properties
            (rootIndex().column()             == other->rootIndex().column()) &&
            (rootIndex().row()                == other->rootIndex().row()) &&
            (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
            (antiAliasing()                   == other->antiAliasing()) &&
            (percentMode()                    == other->percentMode()) &&
            (datasetDimension()               == other->datasetDimension());
}
```

7.10.2.13 AbstractCoordinatePlane ∗ AbstractDiagram::coordinatePlane () const `[inherited]`

The coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. By default this is a CartesianCoordinatePlane.

Returns:

The coordinate plane associated with the diagram.

Definition at line 226 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractCartesianDiagram::layoutPlanes(), KDChart::PolarDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), polarCoordinatePlane(), and KDChart::AbstractCartesianDiagram::setCoordinatePlane().

```
227 {
228     return d->plane;
229 }
```

7.10.2.14 const QPair< QPointF, QPointF > AbstractDiagram::dataBoundaries () const `[inherited]`

Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

This method returns a chached result of calculations done by calculateDataBoundaries. Classes derived from AbstractDiagram must implement the calculateDataBoundaries function, to specify their own way of calculating the data boundaries. If derived classes want to force recalculation of the data boundaries, they can call setDataBoundariesDirty()

Returned value is in diagram coordinates.

Definition at line 231 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::calculateDataBoundaries(), and d.

Referenced by KDChart::CartesianCoordinatePlane::getRawDataBoundingRectFromDiagrams(), KDChart::PolarCoordinatePlane::layoutDiagrams(), KDChart::LineDiagram::paint(), and KDChart::BarDiagram::paint().

```
232 {
233     if( d->databoundariesDirty ){
234         d->databoundaries = calculateDataBoundaries ();
235         d->databoundariesDirty = false;
236     }
237     return d->databoundaries;
238 }
```

7.10.2.15 void AbstractDiagram::dataChanged (const QModelIndex & *topLeft*, const QModelIndex & *bottomRight*) `[virtual, inherited]`

[reimplemented]

Definition at line 338 of file KDChartAbstractDiagram.cpp.

References d.

```
338 {
339   // We are still too dumb to do intelligent updates...
340   d->databoundariesDirty = true;
341   scheduleDelayedItemsLayout();
342 }
343
344
```

7.10.2.16 void KDChart::AbstractDiagram::dataHidden () `[protected, inherited]`

This signal is emitted, when the hidden status of at least one data cell was (un)set.

7.10.2.17 QList< QBrush > AbstractDiagram::datasetBrushes () const `[inherited]`

The set of dataset brushes currently used, for use in legends, etc.

Note:

Cell-level override brushes, if set, take precedence over the dataset values, so you might need to check these too, in order to find the brush, that is used for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 894 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), KDChart::Legend::datasetCount(), and KDChart::Legend::setBrushesFromDiagram().

```
896                                                                                   {
897         QBrush brush = qVariantValue<QBrush>( attributesModel()->headerData( i, Qt::Vertical, DatasetE
898         ret << brush;
899     }
900
901     return ret;
902 }
903
904 QList<QPen> AbstractDiagram::datasetPens() const
```

7.10.2.18 int AbstractDiagram::datasetDimension () const `[inherited]`

The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

For each dimension it will expect one column of values in the model. If the dimensionality is 1, automatic values will be used for the abscissa.

For example a diagram with the default dimension of 1, will have one column per datapoint (the y values) and will use automatic values for the x axis (1, 2, 3, ... n). If the dimension is 2, the diagram will use the first, (and the third, fifth, etc) columns as X values, and the second, (and the fourth, sixth, etc) column as Y values.

Returns:

The dataset dimension of the diagram.

Definition at line 942 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::calculateDataBoundaries(), KDChart::LineDiagram::getCellValues(), KDChart::CartesianCoordinatePlane::getDataDimensionsList(), KDChart::LineDiagram::paint(), and KDChart::LineDiagram::setType().

```
946 {
```

7.10.2.19 QStringList AbstractDiagram::datasetLabels () const `[inherited]`

The set of dataset labels currently displayed, for use in legends, etc.

Returns:

The set of dataset labels currently displayed.

Definition at line 882 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), and KDChart::Legend::datasetCount().

```
                                                 : " << attributesModel()->columnCount(attributesModel
    const int columnCount = attributesModel()->columnCount(attributesModelRootIndex());
    for( int i = datasetDimension()-1; i < columnCount; i += datasetDimension() ){
        //qDebug() << "dataset label: " << attributesModel()->headerData( i, Qt::Horizontal, Qt::Displ
        ret << attributesModel()->headerData( i, Qt::Horizontal, Qt::DisplayRole ).toString();
    }
    return ret;
}

QList<QBrush> AbstractDiagram::datasetBrushes() const
```

7.10.2.20 QList< MarkerAttributes > AbstractDiagram::datasetMarkers () const `[inherited]`

The set of dataset markers currently used, for use in legends, etc.

Note:

Cell-level override markers, if set, take precedence over the dataset values, so you might need to check these too, in order to find the marker, that is shown for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 917 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
                                                                                  {
        DataValueAttributes a =
            qVariantValue<DataValueAttributes>( attributesModel()->headerData( i, Qt::Vertical, DataVa
        const MarkerAttributes &ma = a.markerAttributes();
        ret << ma;
    }
    return ret;
}

bool AbstractDiagram::checkInvariants( bool justReturnTheStatus ) const
```

7.10.2.21 QList< QPen > AbstractDiagram::datasetPens () const `[inherited]`

The set of dataset pens currently used, for use in legends, etc.

Note:

Cell-level override pens, if set, take precedence over the dataset values, so you might need to check these too, in order to find the pens, that is used for a single cell.

Returns:

The current set of dataset pens.

Definition at line 906 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
                                                                                  {
        QPen pen = qVariantValue<QPen>( attributesModel()->headerData( i, Qt::Vertical, DatasetPenRole
        ret << pen;
    }
    return ret;
}

QList<MarkerAttributes> AbstractDiagram::datasetMarkers() const
```

7.10.2.22 DataValueAttributes AbstractDiagram::dataValueAttributes (const QModelIndex & *index*) const `[inherited]`

Retrieve the DataValueAttributes for the given index.

This will fall back automatically to what was set at dataset or model level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the attributes for.

Returns:

The DataValueAttributes for the given index.

Definition at line 427 of file KDChartAbstractDiagram.cpp.

```
{
```

7.10.2.23 DataValueAttributes AbstractDiagram::dataValueAttributes (int *column*) const `[inherited]`

Retrieve the DataValueAttributes for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the attributes for.

Returns:

The DataValueAttributes for the given dataset.

Definition at line 420 of file KDChartAbstractDiagram.cpp.

```
{
```

7.10.2.24 DataValueAttributes AbstractDiagram::dataValueAttributes () const `[inherited]`

Retrieve the DataValueAttributes speficied globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The global DataValueAttributes.

Definition at line 414 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::AbstractDiagram::paintDataValueText(), and KDChart::AbstractDiagram::paintMarker().

```
419 {
```

7.10.2.25 void AbstractDiagram::doItemsLayout () `[virtual, inherited]`

[reimplemented]

Definition at line 329 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::AbstractDiagram::update().

```
329                     {
330         d->plane->layoutDiagrams();
331         update();
332     }
333     QAbstractItemView::doItemsLayout();
334 }
335 
336 void AbstractDiagram::dataChanged( const QModelIndex &topLeft,
```

7.10.2.26 int AbstractDiagram::horizontalOffset () const `[virtual, inherited]`

[reimplemented]

Definition at line 839 of file KDChartAbstractDiagram.cpp.

```
841 { return 0; }
```

7.10.2.27 QModelIndex AbstractDiagram::indexAt (const QPoint & *point*) const `[virtual, inherited]`

[reimplemented]

Definition at line 833 of file KDChartAbstractDiagram.cpp.

```
835 { return QModelIndex(); }
```

7.10.2.28 bool AbstractDiagram::isHidden (const QModelIndex & *index*) const `[inherited]`

Retrieve the hidden status for the given index.

This will fall back automatically to what was set at dataset or diagram level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the hidden status for.

Returns:

The hidden status for the given index.

Definition at line 386 of file KDChartAbstractDiagram.cpp.

7.10.2.29 bool AbstractDiagram::isHidden (int *column*) const `[inherited]`

Retrieve the hidden status for the given dataset.

This will fall back automatically to what was set at diagram level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the hidden status for.

Returns:

The hidden status for the given dataset.

Definition at line 379 of file KDChartAbstractDiagram.cpp.

```
385 {
```

7.10.2.30 bool AbstractDiagram::isHidden () const `[inherited]`

Retrieve the hidden status speficied globally.

This will fall back automatically to the default settings (= not hidden), if there are no specific settings.

Returns:

The global hidden status.

Definition at line 373 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), KDChart::LineDiagram::paint(), and KDChart::LineDiagram::valueForCellTesting().

```
378 {
```

7.10.2.31 bool AbstractDiagram::isIndexHidden (const QModelIndex & *index*) const `[virtual, inherited]`

[reimplemented]

Definition at line 845 of file KDChartAbstractDiagram.cpp.

```
847 {}
```

7.10.2.32 QStringList AbstractDiagram::itemRowLabels () const `[inherited]`

The set of item row labels currently displayed, for use in Abscissa axes, etc.

Returns:

The set of item row labels currently displayed.

Definition at line 870 of file KDChartAbstractDiagram.cpp.

```
                                                  : " << attributesModel()->rowCount(attributesModelRo
    const int rowCount = attributesModel()->rowCount(attributesModelRootIndex());
    for( int i = 0; i < rowCount; ++i ){
        //qDebug() << "item row label: " << attributesModel()->headerData( i, Qt::Vertical, Qt::Displa
        ret << attributesModel()->headerData( i, Qt::Vertical, Qt::DisplayRole ).toString();
    }
    return ret;
}

QStringList AbstractDiagram::datasetLabels() const
```

7.10.2.33 void KDChart::AbstractDiagram::modelsChanged () `[protected, inherited]`

This signal is emitted, when either the model or the AttributesModel is replaced.

Referenced by KDChart::AbstractDiagram::setAttributesModel(), and KDChart::AbstractDiagram::setModel().

7.10.2.34 QModelIndex AbstractDiagram::moveCursor (CursorAction *cursorAction*, Qt::KeyboardModifiers *modifiers*) `[virtual, inherited]`

[reimplemented]

Definition at line 836 of file KDChartAbstractDiagram.cpp.

```
{ return 0; }
```

7.10.2.35 virtual double KDChart::AbstractPolarDiagram::numberOfGridRings () const `[pure virtual]`

Implemented in KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

7.10.2.36 virtual double KDChart::AbstractPolarDiagram::numberOfValuesPerDataset () const `[pure virtual]`

Implemented in KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

Referenced by columnCount().

7.10.2.37 virtual void KDChart::AbstractDiagram::paint (PaintContext ∗ *paintContext*) `[pure virtual, inherited]`

Draw the diagram contents to the rectangle and painter, that are passed in as part of the paint context.

Parameters:

paintContext All information needed for painting.

Implemented in KDChart::BarDiagram, KDChart::LineDiagram, KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

7.10.2.38 void AbstractDiagram::paintDataValueText (QPainter ∗ *painter*, const QModelIndex & *index*, const QPointF & *pos*, double *value*) `[inherited]`

Definition at line 474 of file KDChartAbstractDiagram.cpp.

References KDChart::RelativePosition::alignment(), KDChart::TextAttributes::calculatedFont(), d, KDChart::DataValueAttributes::dataLabel(), KDChart::AbstractDiagram::dataValueAttributes(), KDChart::DataValueAttributes::decimalDigits(), KDChart::TextAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::TextAttributes::pen(), KDChart::DataValueAttributes::position(), KDChart::DataValueAttributes::prefix(), KDChart::TextAttributes::rotation(), KDChart::DataValueAttributes::showRepetitiveDataLabels(), KDChart::DataValueAttributes::suffix(), and KDChart::DataValueAttributes::textAttributes().

Referenced by KDChart::RingDiagram::paint(), and KDChart::PolarDiagram::paint().

```
{
    // paint one data series
    const DataValueAttributes a( dataValueAttributes(index) );
    if ( !a.isVisible() ) return;

    // handle decimal digits
    int decimalDigits = a.decimalDigits();
    int decimalPos = QString::number(  value ).indexOf( QLatin1Char( '.' ) );
    QString roundedValue;
    if ( a.dataLabel().isNull() ) {
        if ( decimalPos > 0 && value != 0 )
            roundedValue =  roundValues ( value, decimalPos, decimalDigits );
        else
            roundedValue = QString::number(  value );
    } else
        roundedValue = a.dataLabel();
        // handle prefix and suffix
    if ( !a.prefix().isNull() )
        roundedValue.prepend( a.prefix() );

    if ( !a.suffix().isNull() )
        roundedValue.append( a.suffix() );

    const TextAttributes ta( a.textAttributes() );
    // FIXME draw the non-text bits, background, etc
    if ( ta.isVisible() ) {

        QPointF pt( pos );
        /* for debugging:
        PainterSaver painterSaver( painter );
        painter->setPen( Qt::black );
        painter->drawLine( pos - QPointF( 1,1), pos + QPointF( 1,1) );
        painter->drawLine( pos - QPointF(-1,1), pos + QPointF(-1,1) );
        */

        // adjust the text start point position, if alignment is not Bottom/Left
        const RelativePosition relPos( a.position( value >= 0.0 ) );
        const Qt::Alignment alignBottomLeft = Qt::AlignBottom | Qt::AlignLeft;
        const QFont calculatedFont( ta.calculatedFont( d->plane, KDChartEnums::MeasureOrientationMinim
        //qDebug() << "calculatedFont's point size:" << calculatedFont.pointSizeF();
        if( (relPos.alignment() & alignBottomLeft) != alignBottomLeft ){
            const QRectF boundRect(
                    d->cachedFontMetrics( calculatedFont, this )->boundingRect( roundedValue ) );
            if( relPos.alignment() & Qt::AlignRight )
                pt.rx() -= boundRect.width();
            else if( relPos.alignment() & Qt::AlignHCenter )
                pt.rx() -= 0.5 * boundRect.width();

            if( relPos.alignment() & Qt::AlignTop )
                pt.ry() += boundRect.height();
            else if( relPos.alignment() & Qt::AlignVCenter )
                pt.ry() += 0.5 * boundRect.height();
        }

        // FIXME draw the non-text bits, background, etc

        if ( a.showRepetitiveDataLabels() ||
             pos.x() <= d->lastX ||
             d->lastRoundedValue != roundedValue ) {
            d->lastRoundedValue = roundedValue;
            d->lastX = pos.x();

            PainterSaver painterSaver( painter );
            painter->setPen( ta.pen() );
            painter->setFont( calculatedFont );
            painter->translate( pt );
            painter->rotate( ta.rotation() );
            painter->drawText( QPointF(0, 0), roundedValue );
        }
    }
}
```

7.10.2.39 void AbstractDiagram::paintDataValueTexts (QPainter * *painter*) [protected, virtual, inherited]

Definition at line 576 of file KDChartAbstractDiagram.cpp.

```
                                                                               {
       for ( int j=0; j< rowCount; ++j ) {
           const QModelIndex index = model()->index( j, i, rootIndex() );
           double value = model()->data( index ).toDouble();
           const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
           paintDataValueText( painter, index, pos, value );
       }
    }
}
```

7.10.2.40 void AbstractDiagram::paintMarker (QPainter * *painter*, const QModelIndex & *index*, const QPointF & *pos*) [inherited]

Definition at line 592 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::brush(), KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractDiagram::dataValueAttributes(), KDChart::MarkerAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::DataValueAttributes::markerAttributes(), KDChart::MarkerAttributes::markerColor(), KDChart::MarkerAttributes::markerSize(), KDChart::AbstractDiagram::paintMarker(), and KDChart::MarkerAttributes::pen().

```
{

    if ( !checkInvariants() ) return;
    DataValueAttributes a = dataValueAttributes(index);
    if ( !a.isVisible() ) return;
    const MarkerAttributes &ma = a.markerAttributes();
    if ( !ma.isVisible() ) return;

    PainterSaver painterSaver( painter );
    QSizeF maSize( ma.markerSize() );
    QBrush indexBrush( brush( index ) );
    QPen indexPen( ma.pen() );
    if ( ma.markerColor().isValid() )
        indexBrush.setColor( ma.markerColor() );

    paintMarker( painter, ma, indexBrush, indexPen, pos, maSize );
}


```

7.10.2.41 void AbstractDiagram::paintMarker (QPainter ∗ *painter*, const MarkerAttributes & *markerAttributes*, const QBrush & *brush*, const QPen &, const QPointF & *point*, const QSizeF & *size*) `[virtual, inherited]`

Definition at line 614 of file KDChartAbstractDiagram.cpp.

References KDChart::MarkerAttributes::markerStyle().

Referenced by KDChart::MarkerLayoutItem::paintIntoRect(), and KDChart::AbstractDiagram::paintMarker().

```
{

    const QPen oldPen( painter->pen() );
    // Pen is used to paint 4Pixels - 1 Pixel - Ring and FastCross types.
    // make sure to use the brush color - see above in those cases.
    const bool isFourPixels = (markerAttributes.markerStyle() == MarkerAttributes::Marker4Pixels);
    if( isFourPixels || (markerAttributes.markerStyle() == MarkerAttributes::Marker1Pixel) ){
        // for high-performance point charts with tiny point markers:
        painter->setPen( QPen( brush.color().light() ) );
        if( isFourPixels ){
            const qreal x = pos.x();
            const qreal y = pos.y();
            painter->drawLine( QPointF(x-1.0,y-1.0),
                               QPointF(x+1.0,y-1.0) );
            painter->drawLine( QPointF(x-1.0,y),
                               QPointF(x+1.0,y) );
            painter->drawLine( QPointF(x-1.0,y+1.0),
                               QPointF(x+1.0,y+1.0) );
        }
        painter->drawPoint( pos );
    }else{
        PainterSaver painterSaver( painter );
        // we only a solid line surrounding the markers
        QPen painterPen( pen );
        painterPen.setStyle( Qt::SolidLine );
        painter->setPen( painterPen );
        painter->setBrush( brush );
        painter->setRenderHint ( QPainter::Antialiasing );
        painter->translate( pos );
        switch ( markerAttributes.markerStyle() ) {
            case MarkerAttributes::MarkerCircle:
                painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                            maSize.height(), maSize.width()) );
                break;
            case MarkerAttributes::MarkerSquare:
                {
                    QRectF rect( 0 - maSize.width()/2, 0 - maSize.height()/2,
                                maSize.width(), maSize.height() );
                    painter->drawRect( rect );
                    painter->fillRect( rect, brush.color() );
                    break;
                }
            case MarkerAttributes::MarkerDiamond:
                {
                    QVector <QPointF > diamondPoints;
                    QPointF top, left, bottom, right;
                    top    = QPointF( 0, 0 - maSize.height()/2 );
                    left   = QPointF( 0 - maSize.width()/2, 0 );
                    bottom = QPointF( 0, maSize.height()/2 );
                    right  = QPointF( maSize.width()/2, 0 );
                    diamondPoints << top << left << bottom << right;
                    painter->drawPolygon( diamondPoints );
                    break;
                }
            // both handled on top of the method:
            case MarkerAttributes::Marker1Pixel:
            case MarkerAttributes::Marker4Pixels:
                    break;
            case MarkerAttributes::MarkerRing:
                {
                    painter->setPen( QPen( brush.color() ) );
                    painter->setBrush( Qt::NoBrush );
                    painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                                        maSize.height(), maSize.width()) );
                    break;
                }
            case MarkerAttributes::MarkerCross:
                {
                    QRectF rect( maSize.width()*-0.5, maSize.height()*-0.2,
                                 maSize.width(), maSize.height()*0.4 );
                    painter->drawRect( rect );
                    rect.setTopLeft(QPointF( maSize.width()*-0.2, maSize.height()*-0.5 ));
                    rect.setSize(QSizeF( maSize.width()*0.4, maSize.height() ));
                    painter->drawRect( rect );
                    break;
                }
            case MarkerAttributes::MarkerFastCross:
                {
                    QPointF left, right, top, bottom;
                    left  = QPointF( -maSize.width()/2, 0 );
                    right = QPointF( maSize.width()/2, 0 );
                    top   = QPointF( 0, -maSize.height()/2 );
                    bottom= QPointF( 0, maSize.height()/2 );
                    painter->setPen( QPen( brush.color() ) );
                    painter->drawLine( left, right );
                    painter->drawLine(  top, bottom );
                    break;
                }
            default:
                Q_ASSERT_X ( false, "paintMarkers()",
                            "Type item does not match a defined Marker Type." );
        }
    }
    painter->setPen( oldPen );
}

void AbstractDiagram::paintMarkers( QPainter* painter )
```

7.10.2.42 void AbstractDiagram::paintMarkers (QPainter ∗ *painter*) `[protected, virtual, inherited]`

Definition at line 716 of file KDChartAbstractDiagram.cpp.

```
                                                                                 {
       for ( int j=0; j< rowCount; ++j ) {
           const QModelIndex index = model()->index( j, i, rootIndex() );
           double value = model()->data( index ).toDouble();
           const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
           paintMarker( painter, index, pos );
       }
    }
}


```

7.10.2.43 QPen AbstractDiagram::pen (const QModelIndex & *index*) const `[inherited]`

Retrieve the pen to be used, for painting the datapoint at the given index in the model.

Parameters:

index The index of the datapoint in the model.

Returns:

The pen to use for painting.

Definition at line 770 of file KDChartAbstractDiagram.cpp.

```
{
```

7.10.2.44 QPen AbstractDiagram::pen (int *dataset*) const `[inherited]`

Retrieve the pen to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the pen for.

Returns:

The pen to use for painting.

Definition at line 762 of file KDChartAbstractDiagram.cpp.

```
{
```

7.10.2.45 QPen AbstractDiagram::pen () const `[inherited]`

Retrieve the pen to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The pen to use for painting.

Definition at line 756 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::PieDiagram::paint(), and KDChart::LineDiagram::paint().

```
761 {
```

7.10.2.46 bool AbstractDiagram::percentMode () const `[inherited]`

Definition at line 468 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::CartesianCoordinatePlane::getDataDimensionsList().

7.10.2.47 const PolarCoordinatePlane $*$ AbstractPolarDiagram::polarCoordinatePlane () const

Definition at line 55 of file KDChartAbstractPolarDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane().

Referenced by KDChart::PieDiagram::paint().

```
56 {
57     return dynamic_cast<const PolarCoordinatePlane*>( coordinatePlane() );
58 }
```

7.10.2.48 void KDChart::AbstractDiagram::propertiesChanged () `[protected, inherited]`

Emitted upon change of a property of the Diagram.

Referenced by KDChart::LineDiagram::resetLineAttributes(), KDChart::AbstractDiagram::setDataValueAttributes(), KDChart::LineDiagram::setLineAttributes(), KDChart::LineDiagram::setThreeDLineAttributes(), and KDChart::LineDiagram::setType().

7.10.2.49 virtual void KDChart::AbstractDiagram::resize (const QSizeF & *area*) `[pure virtual, inherited]`

Called by the widget's sizeEvent.

Adjust all internal structures, that are calculated, dependending on the size of the widget.

Parameters:

area

Implemented in KDChart::BarDiagram, KDChart::LineDiagram, KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

7.10.2.50 void AbstractDiagram::scrollTo (const QModelIndex & *index*, ScrollHint *hint* = EnsureVisible) `[virtual, inherited]`

[reimplemented]

Definition at line 830 of file KDChartAbstractDiagram.cpp.

```
832 { return QModelIndex(); }
```

7.10.2.51 void AbstractDiagram::setAllowOverlappingDataValueTexts (bool *allow*) `[inherited]`

Set whether data value labels are allowed to overlap.

Parameters:

allow True means that overlapping labels are allowed.

Definition at line 440 of file KDChartAbstractDiagram.cpp.

References d.

```
445 {
```

7.10.2.52 void AbstractDiagram::setAntiAliasing (bool *enabled*) `[inherited]`

Set whether anti-aliasing is to be used while rendering this diagram.

Parameters:

enabled True means that AA is enabled.

Definition at line 451 of file KDChartAbstractDiagram.cpp.

References d.

```
456 {
```

7.10.2.53 void AbstractDiagram::setAttributesModel (AttributesModel ∗ *model*) `[virtual, inherited]`

Associate an AttributesModel with this diagram.

Note that the diagram does _not_ take ownership of the AttributesModel. This should thus only be used with AttributesModels that have been explicitely created by the user, and are owned by her. Setting an AttributesModel that is internal to another diagram is an error.

Correct:

```
AttributesModel *am = new AttributesModel( model, 0 );
diagram1->setAttributesModel( am );
diagram2->setAttributesModel( am );
```

Wrong:

```
diagram1->setAttributesModel( diagram2->attributesModel() );
```

Parameters:

model The AttributesModel to use for this diagram.

See also:

AttributesModel, usesExternalAttributesModel

Definition at line 261 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::AbstractDiagram::modelsChanged().

```
{
    if( amodel->sourceModel() != model() ) {
        qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
                 "Trying to set an attributesmodel which works on a different "
                 "model than the diagram.");
        return;
    }
    if( qobject_cast<PrivateAttributesModel*>(amodel) ) {
        qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
                 "Trying to set an attributesmodel that is private to another diagram.");
        return;
    }
    d->setAttributesModel(amodel);
    scheduleDelayedItemsLayout();
    d->databoundariesDirty = true;
    emit modelsChanged();
}
```

7.10.2.54 void AbstractDiagram::setAttributesModelRootIndex (const QModelIndex & *idx*) `[protected, inherited]`

Definition at line 301 of file KDChartAbstractDiagram.cpp.

References d.

7.10.2.55 void AbstractDiagram::setBrush (const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting all datasets in the model.

Parameters:

brush The brush to use.

Definition at line 786 of file KDChartAbstractDiagram.cpp.

```
{
```

7.10.2.56 void AbstractDiagram::setBrush (int *dataset*, const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting the given dataset.

Parameters:

dataset The dataset's column in the model.

pen The brush to use.

Definition at line 793 of file KDChartAbstractDiagram.cpp.

```
801 {
```

7.10.2.57 void AbstractDiagram::setBrush (const QModelIndex & *index*, const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting the datapoint at the given index.

Parameters:

index The datapoint's index in the model.

brush The brush to use.

Definition at line 778 of file KDChartAbstractDiagram.cpp.

```
785 {
```

7.10.2.58 void AbstractDiagram::setCoordinatePlane (AbstractCoordinatePlane ∗ *plane*) `[virtual, inherited]`

Set the coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. The chart takes ownership.

Returns:

The coordinate plane associated with the diagram.

Reimplemented in KDChart::AbstractCartesianDiagram.

Definition at line 324 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractCoordinatePlane::addDiagram(), KDChart::AbstractCartesianDiagram::setCoordinatePlane(), and KDChart::AbstractCoordinatePlane::takeDiagram().

```
328 {
```

7.10.2.59 void AbstractDiagram::setDataBoundariesDirty () const `[protected, inherited]`

Definition at line 240 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::BarDiagram::setThreeDBarAttributes(), KDChart::LineDiagram::setThreeDLineAttributes(), KDChart::LineDiagram::setType(), and KDChart::BarDiagram::setType().

```
241 {
242     d->databoundariesDirty = true;
243 }
```

7.10.2.60 void AbstractDiagram::setDatasetDimension (int *dimension*) `[inherited]`

Sets the dataset dimension of the diagram.

See also:
datasetDimension.

Parameters:
dimension

Definition at line 947 of file KDChartAbstractDiagram.cpp.

References d.

```
954 {
```

7.10.2.61 void AbstractDiagram::setDataValueAttributes (const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for all datapoints in the model.

Parameters:
a The attributes to set.

Definition at line 434 of file KDChartAbstractDiagram.cpp.

References d.

```
439 {
```

7.10.2.62 void AbstractDiagram::setDataValueAttributes (int *dataset*, const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for the given dataset.

Parameters:
dataset The dataset to set the attributes for.

a The attributes to set.

Definition at line 406 of file KDChartAbstractDiagram.cpp.

References d.

```
413 {
```

7.10.2.63 void AbstractDiagram::setDataValueAttributes (const QModelIndex & *index*, const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for the given index.

Parameters:

index The datapoint to set the attributes for.

a The attributes to set.

Definition at line 395 of file KDChartAbstractDiagram.cpp.

References d, KDChart::DataValueLabelAttributesRole, and KDChart::AbstractDiagram::propertiesChanged().

```
395 {
396     d->attributesModel->setData(
397         d->attributesModel->mapFromSource( index ),
398         qVariantFromValue( a ),
399         DataValueLabelAttributesRole );
400     emit propertiesChanged();
401 }
402
403
```

7.10.2.64 void AbstractDiagram::setHidden (bool *hidden*) `[inherited]`

Hide (or unhide, resp.) all datapoints in the model.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

hidden The hidden status to set.

Definition at line 365 of file KDChartAbstractDiagram.cpp.

References d.

```
372 {
```

7.10.2.65 void AbstractDiagram::setHidden (int *column*, bool *hidden*) `[inherited]`

Hide (or unhide, resp.) a dataset.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

dataset The dataset to set the hidden status for.

hidden The hidden status to set.

Definition at line 356 of file KDChartAbstractDiagram.cpp.

References d.

```
364 {
```

7.10.2.66 void AbstractDiagram::setHidden (const QModelIndex & *index*, bool *hidden*) `[inherited]`

Hide (or unhide, resp.) a data cell.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

index The datapoint to set the hidden status for.

hidden The hidden status to set.

Definition at line 347 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::DataHiddenRole.

```
355 {
```

7.10.2.67 void AbstractDiagram::setModel (QAbstractItemModel ∗ *model*) `[virtual, inherited]`

Associate a model with the diagram.

Definition at line 245 of file KDChartAbstractDiagram.cpp.

References d, KDChart::AttributesModel::initFrom(), and KDChart::AbstractDiagram::modelsChanged().

```
246 {
247   QAbstractItemView::setModel( newModel );
248   AttributesModel* amodel = new PrivateAttributesModel( newModel, this );
249   amodel->initFrom( d->attributesModel );
250   d->setAttributesModel(amodel);
251   scheduleDelayedItemsLayout();
252   d->databoundariesDirty = true;
253   emit modelsChanged();
254 }
```

7.10.2.68 void AbstractDiagram::setPen (const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting all datasets in the model.

Parameters:

pen The pen to use.

Definition at line 740 of file KDChartAbstractDiagram.cpp.

```
746 {
```

7.10.2.69 void AbstractDiagram::setPen (int *dataset*, const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting the given dataset.

Parameters:

dataset The dataset's row in the model.

pen The pen to use.

Definition at line 747 of file KDChartAbstractDiagram.cpp.

```
755 {
```

7.10.2.70 void AbstractDiagram::setPen (const QModelIndex & *index*, const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting the datapoint at the given index.

Parameters:

index The datapoint's index in the model.

pen The pen to use.

Definition at line 732 of file KDChartAbstractDiagram.cpp.

```
739 {
```

7.10.2.71 void AbstractDiagram::setPercentMode (bool *percent*) `[inherited]`

Definition at line 462 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::setType(), and KDChart::BarDiagram::setType().

```
467 {
```

7.10.2.72 void AbstractDiagram::setRootIndex (const QModelIndex & *idx*) `[virtual, inherited]`

Set the root index in the model, where the diagram starts referencing data for display.

[reimplemented]

Definition at line 294 of file KDChartAbstractDiagram.cpp.

References d.

7.10.2.73 void AbstractDiagram::setSelection (const QRect & *rect*, QItemSelectionModel::SelectionFlags *command*) `[virtual, inherited]`

[reimplemented]

Definition at line 848 of file KDChartAbstractDiagram.cpp.

```
850 { return QRegion(); }
```

7.10.2.74 void AbstractDiagram::update () const `[inherited]`

Definition at line 961 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractDiagram::doItemsLayout().

7.10.2.75 void KDChart::AbstractDiagram::useDefaultColors () `[inherited]`

Set the palette to be used, for painting datasets to the default palette.

See also:

KDChart::Palette. FIXME: fold into one usePalette (KDChart::Palette&) method

Definition at line 855 of file KDChartAbstractDiagram.cpp.

References d.

```
859 {
```

7.10.2.76 void KDChart::AbstractDiagram::useRainbowColors () `[inherited]`

Set the palette to be used, for painting datasets to the rainbow palette.

See also:

KDChart::Palette.

Definition at line 865 of file KDChartAbstractDiagram.cpp.

References d.

```
869 {
```

7.10.2.77 bool AbstractDiagram::usesExternalAttributesModel () const `[virtual, inherited]`

Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

See also:
setAttributesModel

Definition at line 280 of file KDChartAbstractDiagram.cpp.

References d.

```
281 {
282     return d->usesExternalAttributesModel();
283 }
```

7.10.2.78 void KDChart::AbstractDiagram::useSubduedColors () `[inherited]`

Set the palette to be used, for painting datasets to the subdued palette.

See also:
KDChart::Palette.

Definition at line 860 of file KDChartAbstractDiagram.cpp.

References d.

```
864 {
```

7.10.2.79 double AbstractDiagram::valueForCell (int *row*, int *column*) const `[protected, inherited]`

Helper method, retrieving the data value (DisplayRole) for a given row and column.

Parameters:

row The row to query.

column The column to query.

Returns:
The value of the display role at the given row and column as a double.

Definition at line 955 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::attributesModelRootIndex(), and d.

Referenced by KDChart::LineDiagram::paint().

```
960 {
```

7.10.2.80 virtual double KDChart::AbstractPolarDiagram::valueTotals () const `[pure virtual]`

Implemented in KDChart::PieDiagram, KDChart::PolarDiagram, and KDChart::RingDiagram.

Referenced by KDChart::PolarCoordinatePlane::layoutDiagrams().

7.10.2.81 int AbstractDiagram::verticalOffset () const `[virtual, inherited]`

[reimplemented]

Definition at line 842 of file KDChartAbstractDiagram.cpp.

```
844 { return true; }
```

7.10.2.82 QRect AbstractDiagram::visualRect (const QModelIndex & *index*) const `[virtual, inherited]`

[reimplemented]

Definition at line 825 of file KDChartAbstractDiagram.cpp.

```
829 {}
```

7.10.2.83 QRegion AbstractDiagram::visualRegionForSelection (const QItemSelection & *selection*) const `[virtual, inherited]`

[reimplemented]

Definition at line 851 of file KDChartAbstractDiagram.cpp.

7.10.3 Member Data Documentation

7.10.3.1 Q_SIGNALS KDChart::AbstractDiagram::__pad0__ `[protected, inherited]`

Definition at line 589 of file KDChartAbstractDiagram.h.

The documentation for this class was generated from the following files:

- KDChartAbstractPolarDiagram.h
- KDChartAbstractPolarDiagram.cpp

7.11 KDChart::AbstractProxyModel Class Reference

`#include <KDChartAbstractProxyModel.h>`

Inheritance diagram for KDChart::AbstractProxyModel:Collaboration diagram for KDChart::AbstractProxyModel:

Public Member Functions

- AbstractProxyModel (QObject ∗parent=0)

  This is basically KDAbstractProxyModel, but only the bits that we really need from it.

- QModelIndex index (int row, int col, const QModelIndex &index) const
- QModelIndex mapFromSource (const QModelIndex &sourceIndex) const
- QModelIndex mapToSource (const QModelIndex &proxyIndex) const
- QModelIndex parent (const QModelIndex &index) const

7.11.1 Constructor & Destructor Documentation

7.11.1.1 KDChart::AbstractProxyModel::AbstractProxyModel (QObject ∗ *parent* = 0) `[explicit]`

This is basically KDAbstractProxyModel, but only the bits that we really need from it.

Definition at line 12 of file KDChartAbstractProxyModel.cpp.

```
  : QAbstractProxyModel(parent) {}
```

7.11.2 Member Function Documentation

7.11.2.1 QModelIndex KDChart::AbstractProxyModel::index (int *row*, int *col*, const QModelIndex & *index*) const

Definition at line 53 of file KDChartAbstractProxyModel.cpp.

References mapFromSource(), and mapToSource().

Referenced by KDChart::AttributesModel::setHeaderData(), and KDChart::AttributesModel::setModelData().

```
{
    Q_ASSERT(sourceModel());
    return mapFromSource(sourceModel()->index( row, col, mapToSource(index) ));
}
```

7.11.2.2 QModelIndex KDChart::AbstractProxyModel::mapFromSource (const QModelIndex & *sourceIndex*) const

Definition at line 23 of file KDChartAbstractProxyModel.cpp.

Referenced by index(), and parent().

```
{
  if ( !sourceIndex.isValid() )
    return QModelIndex();
  //qDebug() << "sourceIndex.model()="<<sourceIndex.model();
  //qDebug() << "model()="<<sourceModel();
  Q_ASSERT( sourceIndex.model() == sourceModel() );

  // Create an index that preserves the internal pointer from the source;
  // this way AbstractProxyModel preserves the structure of the source model
  return createIndex( sourceIndex.row(), sourceIndex.column(), sourceIndex.internalPointer() );
}
```

7.11.2.3 QModelIndex KDChart::AbstractProxyModel::mapToSource (const QModelIndex & *proxyIndex*) const

Definition at line 36 of file KDChartAbstractProxyModel.cpp.

Referenced by KDChart::AttributesModel::columnCount(), KDChart::AttributesModel::data(), index(), parent(), KDChart::AttributesModel::rowCount(), and KDChart::AttributesModel::setData().

```
{
  if ( !proxyIndex.isValid() )
    return QModelIndex();
  Q_ASSERT( proxyIndex.model() == this );
  // So here we need to create a source index which holds that internal pointer.
  // No way to pass it to sourceModel()->index... so we have to do the ugly way:
  QModelIndex sourceIndex;
  KDPrivateModelIndex* hack = reinterpret_cast<KDPrivateModelIndex*>(&sourceIndex);
  hack->r = proxyIndex.row();
  hack->c = proxyIndex.column();
  hack->p = proxyIndex.internalPointer();
  hack->m = sourceModel();
  Q_ASSERT( sourceIndex.isValid() );
  return sourceIndex;
}
```

7.11.2.4 QModelIndex KDChart::AbstractProxyModel::parent (const QModelIndex & *index*) const

Definition at line 59 of file KDChartAbstractProxyModel.cpp.

References mapFromSource(), and mapToSource().

```
{
    Q_ASSERT(sourceModel());
    return mapFromSource(sourceModel()->parent( mapToSource(index) ));
}
```

The documentation for this class was generated from the following files:

- KDChartAbstractProxyModel.h
- KDChartAbstractProxyModel.cpp

7.12 KDChart::AbstractThreeDAttributes Class Reference

`#include <KDChartAbstractThreeDAttributes.h>`

Inheritance diagram for KDChart::AbstractThreeDAttributes:

Public Member Functions

- AbstractThreeDAttributes (const AbstractThreeDAttributes &)
- AbstractThreeDAttributes ()
- double depth () const
- bool isEnabled () const
- bool operator!= (const AbstractThreeDAttributes &other) const
- AbstractThreeDAttributes & operator= (const AbstractThreeDAttributes &)
- bool operator== (const AbstractThreeDAttributes &) const
- void setDepth (double depth)
- void setEnabled (bool enabled)
- double validDepth () const
- virtual ∼AbstractThreeDAttributes ()=0

7.12.1 Constructor & Destructor Documentation

7.12.1.1 AbstractThreeDAttributes::AbstractThreeDAttributes ()

Definition at line 46 of file KDChartAbstractThreeDAttributes.cpp.

```
47     : _d( new Private() )
48 {
49 }
```

7.12.1.2 AbstractThreeDAttributes::AbstractThreeDAttributes (const AbstractThreeDAttributes &)

Definition at line 51 of file KDChartAbstractThreeDAttributes.cpp.

References d.

```
52     : _d( new Private( *r.d ) )
53 {
54 }
```

7.12.1.3 AbstractThreeDAttributes::∼AbstractThreeDAttributes () `[pure virtual]`

Definition at line 66 of file KDChartAbstractThreeDAttributes.cpp.

```
67 {
68     delete _d; _d = 0;
69 }
```

7.12.2 Member Function Documentation

7.12.2.1 double AbstractThreeDAttributes::depth () const

Definition at line 103 of file KDChartAbstractThreeDAttributes.cpp.

References d.

Referenced by operator<<(), operator==(), KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), and KDChart::BarDiagram::paint().

```
104 {
105     return d->depth;
106 }
```

7.12.2.2 bool AbstractThreeDAttributes::isEnabled () const

Definition at line 92 of file KDChartAbstractThreeDAttributes.cpp.

References d.

Referenced by operator<<(), operator==(), KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and validDepth().

```
93 {
94     return d->enabled;
95 }
```

7.12.2.3 bool KDChart::AbstractThreeDAttributes::operator!= (const AbstractThreeDAttributes & *other*) const

Definition at line 57 of file KDChartAbstractThreeDAttributes.h.

```
57 { return !operator==(other); }
```

7.12.2.4 AbstractThreeDAttributes & AbstractThreeDAttributes::operator= (const AbstractThreeDAttributes &)

Definition at line 56 of file KDChartAbstractThreeDAttributes.cpp.

References d.

```
57 {
58     if( this == &r )
59         return *this;
60
61     *d = *r.d;
62
63     return *this;
64 }
```

7.12.2.5 bool AbstractThreeDAttributes::operator== (const AbstractThreeDAttributes &) const

Definition at line 72 of file KDChartAbstractThreeDAttributes.cpp.

References depth(), and isEnabled().

Referenced by KDChart::ThreeDPieAttributes::operator==(), KDChart::ThreeDLineAttributes::operator==(), and KDChart::ThreeDBarAttributes::operator==().

```
{
    if( isEnabled() == r.isEnabled() &&
        depth() == r.depth() )
        return true;
    else
        return false;
}
```

7.12.2.6 void AbstractThreeDAttributes::setDepth (double *depth*)

Definition at line 97 of file KDChartAbstractThreeDAttributes.cpp.

References d.

```
{
    d->depth = depth;
}
```

7.12.2.7 void AbstractThreeDAttributes::setEnabled (bool *enabled*)

Definition at line 87 of file KDChartAbstractThreeDAttributes.cpp.

References d.

```
{
    d->enabled = enabled;
}
```

7.12.2.8 double AbstractThreeDAttributes::validDepth () const

Definition at line 109 of file KDChartAbstractThreeDAttributes.cpp.

References d, and isEnabled().

Referenced by KDChart::LineDiagram::threeDItemDepth(), and KDChart::BarDiagram::threeDItemDepth().

```
{
    return isEnabled() ? d->depth : 0.0;
}
```

The documentation for this class was generated from the following files:

- KDChartAbstractThreeDAttributes.h
- KDChartAbstractThreeDAttributes.cpp

7.13 KDChart::AttributesModel Class Reference

`#include <KDChartAttributesModel.h>`

Inheritance diagram for KDChart::AttributesModel:Collaboration diagram for KDChart::AttributesModel:

Public Types

- enum PaletteType {
  PaletteTypeDefault = 0,
  PaletteTypeRainbow = 1,
  PaletteTypeSubdued = 2 }

Public Member Functions

- AttributesModel (QAbstractItemModel ∗model, QObject ∗parent=0)
- int columnCount (const QModelIndex &) const

  [reimplemented]

- bool compare (const AttributesModel ∗other) const
- bool compareAttributes (int role, const QVariant &a, const QVariant &b) const
- QVariant data (const QModelIndex &, int role=Qt::DisplayRole) const

  [reimplemented]

- QVariant data (int column, int role) const

  Returns the data that were specified at per column level, or the globally set data, or the default data, or QVariant().

- QVariant data (int role) const

  Returns the data that were specified at global level, or the default data, or QVariant().

- QVariant headerData (int section, Qt::Orientation orientation, int role=Qt::DisplayRole) const

  [reimplemented]

- QModelIndex index (int row, int col, const QModelIndex &index) const
- void initFrom (const AttributesModel ∗other)
- bool isKnownAttributesRole (int role) const

  Returns whether the given role corresponds to one of the known internally used ones.

- QModelIndex mapFromSource (const QModelIndex &sourceIndex) const
- QModelIndex mapToSource (const QModelIndex &proxyIndex) const
- QVariant modelData (int role) const
- PaletteType paletteType () const
- QModelIndex parent (const QModelIndex &index) const
- bool resetData (const QModelIndex &index, int role=Qt::DisplayRole)

  Remove any explicit attributes settings that might have been specified before.

- bool resetHeaderData (int section, Qt::Orientation orientation, int role=Qt::DisplayRole)

  Remove any explicit attributes settings that might have been specified before.

- int rowCount (const QModelIndex &) const

  [reimplemented]

- bool setData (const QModelIndex &index, const QVariant &value, int role=Qt::DisplayRole)

  [reimplemented]

- bool setHeaderData (int section, Qt::Orientation orientation, const QVariant &value, int role=Qt::DisplayRole)

  [reimplemented]

- bool setModelData (const QVariant value, int role)
- void setPaletteType (PaletteType type)

  Sets the palettetype used by this attributesmodel.

- void setSourceModel (QAbstractItemModel ∗sourceModel)

  [reimplemented]

- ∼AttributesModel ()

Public Attributes

- Q_SIGNALS __pad0__: void attributesChanged(const QModelIndex&

Protected Member Functions

- const QMap< int, QMap< int, QMap< int, QVariant > > > dataMap () const

  needed for serialization

- const QMap< int, QMap< int, QVariant > > horizontalHeaderDataMap () const

  needed for serialization

- const QMap< int, QVariant > modelDataMap () const

  needed for serialization

- void setDataMap (const QMap< int, QMap< int, QMap< int, QVariant > > > map)

  needed for serialization

- void setHorizontalHeaderDataMap (const QMap< int, QMap< int, QVariant > > map)

  needed for serialization

- void setModelDataMap (const QMap< int, QVariant > map)

  needed for serialization

- void setVerticalHeaderDataMap (const QMap< int, QMap< int, QVariant > > map)

  needed for serialization

- const QMap< int, QMap< int, QVariant > > verticalHeaderDataMap () const

  needed for serialization

7.13.1 Member Enumeration Documentation

7.13.1.1 enum KDChart::AttributesModel::PaletteType

Enumeration values:

- ***PaletteTypeDefault***
- ***PaletteTypeRainbow***
- ***PaletteTypeSubdued***

Definition at line 47 of file KDChartAttributesModel.h.

```
47                      {
48         PaletteTypeDefault = 0,
49         PaletteTypeRainbow = 1,
50         PaletteTypeSubdued = 2
51     };
```

7.13.2 Constructor & Destructor Documentation

7.13.2.1 AttributesModel::AttributesModel (QAbstractItemModel ∗ *model*, QObject ∗ *parent* = 0) `[explicit]`

Definition at line 56 of file KDChartAttributesModel.cpp.

References setSourceModel().

```
57   : AbstractProxyModel( parent ),
58     mPaletteType( PaletteTypeDefault )
59 {
60   setSourceModel(model);
61 }
```

7.13.2.2 AttributesModel::∼AttributesModel ()

Definition at line 63 of file KDChartAttributesModel.cpp.

```
64 {
65 }
```

7.13.3 Member Function Documentation

7.13.3.1 int AttributesModel::columnCount (const QModelIndex &) const

[reimplemented]

Definition at line 503 of file KDChartAttributesModel.cpp.

References KDChart::AbstractProxyModel::mapToSource().

Referenced by setModelData().

```
504 {
505     Q_ASSERT(sourceModel());
506     return sourceModel()->columnCount( mapToSource(index) );
507 }
```

7.13.3.2 bool AttributesModel::compare (const AttributesModel ∗ *other*) const

Definition at line 79 of file KDChartAttributesModel.cpp.

```
{
    if( other == this ) return true;
    if( ! other ){
        //qDebug() << "AttributesModel::compare() cannot compare to Null pointer";
        return false;
    }

    {
        if (mDataMap.count() != other->mDataMap.count()){
            //qDebug() << "AttributesModel::compare() dataMap have different sizes";
            return false;
        }
        QMap<int, QMap<int, QMap<int, QVariant> > >::const_iterator itA = mDataMap.constBegin();
        QMap<int, QMap<int, QMap<int, QVariant> > >::const_iterator itB = other->mDataMap.constBegin();
        while (itA != mDataMap.constEnd()) {
            if ((*itA).count() != (*itB).count()){
                //qDebug() << "AttributesModel::compare() dataMap/map have different sizes";
                return false;
            }
            QMap<int, QMap<int, QVariant> >::const_iterator it2A = (*itA).constBegin();
            QMap<int, QMap<int, QVariant> >::const_iterator it2B = (*itB).constBegin();
            while (it2A != itA->constEnd()) {
                if ((*it2A).count() != (*it2B).count()){
                    //qDebug() << "AttributesModel::compare() dataMap/map/map have different sizes:"
                    //        << (*it2A).count() << (*it2B).count();
                    return false;
                }
                QMap<int, QVariant>::const_iterator it3A = (*it2A).constBegin();
                QMap<int, QVariant>::const_iterator it3B = (*it2B).constBegin();
                while (it3A != it2A->constEnd()) {
                    if ( it3A.key() != it3B.key() ){
                        //qDebug( "AttributesModel::compare()\n"
                        //        "   dataMap[%i, %i] values have different types.  A: %x  B: %x",
                        //        itA.key(), it2A.key(), it3A.key(), it3B.key());
                        return false;
                    }
                    if ( ! compareAttributes( it3A.key(), it3A.value(), it3B.value() ) ){
                        //qDebug( "AttributesModel::compare()\n"
                        //        "   dataMap[%i, %i] values are different. Role: %x", itA.key(), it2A
                        return false;
                    }
                    ++it3A;
                    ++it3B;
                }
                ++it2A;
                ++it2B;
            }
            ++itA;
            ++itB;
        }
    }
    {
        if (mHorizontalHeaderDataMap.count() != other->mHorizontalHeaderDataMap.count()){
            //qDebug() << "AttributesModel::compare() horizontalHeaderDataMap have different sizes";
            return false;
        }
        QMap<int, QMap<int, QVariant> >::const_iterator itA = mHorizontalHeaderDataMap.constBegin();
        QMap<int, QMap<int, QVariant> >::const_iterator itB = other->mHorizontalHeaderDataMap.constBeg
        while (itA != mHorizontalHeaderDataMap.constEnd()) {
            if ((*itA).count() != (*itB).count()){
                //qDebug() << "AttributesModel::compare() horizontalHeaderDataMap/map have different s
                return false;
            }
            QMap<int, QVariant>::const_iterator it2A = (*itA).constBegin();
            QMap<int, QVariant>::const_iterator it2B = (*itB).constBegin();
            while (it2A != itA->constEnd()) {
                if ( it2A.key() != it2B.key() ){
                    //qDebug( "AttributesModel::compare()\n"
                    //        "   horizontalHeaderDataMap[ %i ] values have different types.  A: %x  B
                    //        itA.key(), it2A.key(), it2B.key());
                    return false;
                }
                if ( ! compareAttributes( it2A.key(), it2A.value(), it2B.value() ) ){
                    //qDebug( "AttributesModel::compare()\n"
                    //        "   horizontalHeaderDataMap[ %i ] values are different. Role: %x", itA.k
                    return false;
                }
                ++it2A;
                ++it2B;
            }
            ++itA;
            ++itB;
        }
    }
    {
        if (mVerticalHeaderDataMap.count() != other->mVerticalHeaderDataMap.count()){
            //qDebug() << "AttributesModel::compare() verticalHeaderDataMap have different sizes";
            return false;
        }
        QMap<int, QMap<int, QVariant> >::const_iterator itA = mVerticalHeaderDataMap.constBegin();
        QMap<int, QMap<int, QVariant> >::const_iterator itB = other->mVerticalHeaderDataMap.constBegi
        while (itA != mVerticalHeaderDataMap.constEnd()) {
            if ((*itA).count() != (*itB).count()){
                //qDebug() << "AttributesModel::compare() verticalHeaderDataMap/map have different siz
                return false;
            }
            QMap<int, QVariant>::const_iterator it2A = (*itA).constBegin();
            QMap<int, QVariant>::const_iterator it2B = (*itB).constBegin();
            while (it2A != itA->constEnd()) {
                if ( it2A.key() != it2B.key() ){
                    //qDebug( "AttributesModel::compare()\n"
                    //        "   verticalHeaderDataMap[ %i ] values have different types.  A: %x  B:
                    //        itA.key(), it2A.key(), it2B.key());
                    return false;
                }
                if ( ! compareAttributes( it2A.key(), it2A.value(), it2B.value() ) ){
                    //qDebug( "AttributesModel::compare()\n"
                    //        "   verticalHeaderDataMap[ %i ] values are different. Role: %x", itA.key
                    return false;
                }
                ++it2A;
                ++it2B;
            }
            ++itA;
            ++itB;
        }
    }
    {
        if (mModelDataMap.count() != other->mModelDataMap.count()){
            //qDebug() << "AttributesModel::compare() modelDataMap have different sizes:" << mModelDat
            return false;
        }
        QMap<int, QVariant>::const_iterator itA = mModelDataMap.constBegin();
        QMap<int, QVariant>::const_iterator itB = other->mModelDataMap.constBegin();
        while (itA != mModelDataMap.constEnd()) {
            if ( itA.key() != itB.key() ){
                //qDebug( "AttributesModel::compare()\n"
                //        "   modelDataMap values have different types.  A: %x  B: %x",
                //        itA.key(), itB.key());
                return false;
            }
            if ( ! compareAttributes( itA.key(), itA.value(), itB.value() ) ){
                //qDebug( "AttributesModel::compare()\n"
                //        "   modelDataMap values are different. Role: %x", itA.key() );
                return false;
            }
            ++itA;
            ++itB;
        }
    }
    if (paletteType() != other->paletteType()){
        //qDebug() << "AttributesModel::compare() palette types are different";
        return false;
    }
    return true;
}
```

7.13.3.3 bool AttributesModel::compareAttributes (int *role*, const QVariant & *a*, const QVariant & *b*) const

Definition at line 227 of file KDChartAttributesModel.cpp.

References KDChart::BarAttributesRole, KDChart::DataHiddenRole, KDChart::DatasetBrushRole, KDChart::DatasetPenRole, KDChart::DataValueLabelAttributesRole, KDChart::LineAttributesRole, KDChart::PieAttributesRole, KDChart::ThreeDAttributesRole, KDChart::ThreeDBarAttributesRole, KDChart::ThreeDLineAttributesRole, and KDChart::ThreeDPieAttributesRole.

```
{
    if( isKnownAttributesRole( role ) ){
        switch( role ) {
            case DataValueLabelAttributesRole:
                return (qVariantValue<DataValueAttributes>( a ) ==
                        qVariantValue<DataValueAttributes>( b ));
            case DatasetBrushRole:
                return (qVariantValue<QBrush>( a ) ==
                        qVariantValue<QBrush>( b ));
            case DatasetPenRole:
                return (qVariantValue<QPen>( a ) ==
                        qVariantValue<QPen>( b ));
            case ThreeDAttributesRole:
                // As of yet there is no ThreeDAttributes class,
                // and the AbstractThreeDAttributes class is pure virtual,
                // so we ignore this role for now.
                // (khz, 04.04.2007)
                /*
                return (qVariantValue<ThreeDAttributes>( a ) ==
                        qVariantValue<ThreeDAttributes>( b ));
                */
                break;
            case LineAttributesRole:
                return (qVariantValue<LineAttributes>( a ) ==
                        qVariantValue<LineAttributes>( b ));
            case ThreeDLineAttributesRole:
                return (qVariantValue<ThreeDLineAttributes>( a ) ==
                        qVariantValue<ThreeDLineAttributes>( b ));
            case BarAttributesRole:
                return (qVariantValue<BarAttributes>( a ) ==
                        qVariantValue<BarAttributes>( b ));
            case ThreeDBarAttributesRole:
                return (qVariantValue<ThreeDBarAttributes>( a ) ==
                        qVariantValue<ThreeDBarAttributes>( b ));
            case PieAttributesRole:
                return (qVariantValue<PieAttributes>( a ) ==
265                         qVariantValue<PieAttributes>( b ));
266             case ThreeDPieAttributesRole:
267                 return (qVariantValue<ThreeDPieAttributes>( a ) ==
268                         qVariantValue<ThreeDPieAttributes>( b ));
269             case DataHiddenRole:
270                 return (qVariantValue<bool>( a ) ==
271                         qVariantValue<bool>( b ));
272             default:
273                 Q_ASSERT( false ); // all of our own roles need to be handled
274                 break;
275         }
276     }else{
277         return (a == b);
278     }
279     return true;
280 }
```

7.13.3.4 QVariant AttributesModel::data (const QModelIndex &, int *role* = Qt::DisplayRole) const

[reimplemented]

Definition at line 366 of file KDChartAttributesModel.cpp.

References data(), dataMap(), and KDChart::AbstractProxyModel::mapToSource().

```
367 {
368   //qDebug() << "AttributesModel::data(" << index << role << ")";
369   if( index.isValid() ) {
370     Q_ASSERT( index.model() == this );
371   }
372   QVariant sourceData = sourceModel()->data( mapToSource(index), role );
373   if ( sourceData.isValid() )
374       return sourceData;
375 
376   // check if we are storing a value for this role at this cell index
377   if ( mDataMap.contains( index.column() ) ) {
378       const QMap< int,  QMap< int, QVariant> > &colDataMap = mDataMap[ index.column() ];
379       if ( colDataMap.contains( index.row() ) ) {
380           const QMap<int, QVariant> &dataMap = colDataMap[ index.row() ];
381           if ( dataMap.contains( role ) ) {
382               QVariant v = dataMap[ role ];
383               if( v.isValid() )
384                   return dataMap[ role ];
385           }
386       }
387   }
388   // check if there is something set for the column (dataset), or at global level
389   if( index.isValid() )
390       return data( index.column(), role ); // includes automatic fallback to default
391 
392   return QVariant();
393 }
```

7.13.3.5 QVariant AttributesModel::data (int *column*, int *role*) const

Returns the data that were specified at per column level, or the globally set data, or the default data, or QVariant().

Definition at line 350 of file KDChartAttributesModel.cpp.

References data(), headerData(), and isKnownAttributesRole().

```
{
  if ( isKnownAttributesRole( role ) ) {
      // check if there is something set for the column (dataset)
      QVariant v;
      v = headerData( column, Qt::Vertical, role );

      // check if there is something set at global level
      if ( !v.isValid() )
          v = data( role ); // includes automatic fallback to default
      return v;
  }
  return QVariant();
}
```

7.13.3.6 QVariant AttributesModel::data (int *role*) const

Returns the data that were specified at global level, or the default data, or QVariant().

Definition at line 332 of file KDChartAttributesModel.cpp.

References isKnownAttributesRole(), and modelData().

Referenced by data().

```
{
  if ( isKnownAttributesRole( role ) ) {
      // check if there is something set at global level
      QVariant v = modelData( role );

      // else return the default setting, if any
      if ( !v.isValid() )
          v = defaultsForRole( role );
      return v;
  }
  return QVariant();
}
```

7.13.3.7 const QMap< int, QMap< int, QMap< int, QVariant > > > AttributesModel::dataMap () const `[protected]`

needed for serialization

Definition at line 521 of file KDChartAttributesModel.cpp.

Referenced by data(), headerData(), setData(), and setHeaderData().

```
{
    return mDataMap;
}
```

7.13.3.8 QVariant AttributesModel::headerData (int *section*, Qt::Orientation *orientation*, int *role* = Qt::DisplayRole) const

[reimplemented]

Definition at line 283 of file KDChartAttributesModel.cpp.

References dataMap(), KDChart::DatasetBrushRole, KDChart::Palette::defaultPalette(), KDChart::Palette::getBrush(), modelData(), paletteType(), PaletteTypeDefault, PaletteTypeRainbow, PaletteTypeSubdued, KDChart::Palette::rainbowPalette(), and KDChart::Palette::subduedPalette().

Referenced by data(), KDChart::RingDiagram::paint(), and KDChart::PolarDiagram::paint().

```
{
 QVariant sourceData = sourceModel()->headerData( section, orientation, role );
 if ( sourceData.isValid() ) return sourceData;
 // the source model didn’t have data set, let’s use our stored values
 const QMap<int, QMap<int, QVariant> >& map = orientation == Qt::Horizontal ? mHorizontalHeaderDataMa
 if ( map.contains( section ) ) {
     const QMap<int, QVariant> &dataMap = map[ section ];
     if ( dataMap.contains( role ) ) {
         return dataMap[ role ];
     }
 }

 // Default values if nothing else matches
 switch ( role ) {
 case Qt::DisplayRole:
     return QLatin1String( orientation == Qt::Vertical ?  "Series " : "Item " ) + QString::number( se

 case KDChart::DatasetBrushRole: {
     if ( paletteType() == PaletteTypeSubdued )
         return Palette::subduedPalette().getBrush( section );
     else if ( paletteType() == PaletteTypeRainbow )
         return Palette::rainbowPalette().getBrush( section );
     else if ( paletteType() == PaletteTypeDefault )
         return Palette::defaultPalette().getBrush( section );
     else
         qWarning("Unknown type of fallback palette!");
 }
 case KDChart::DatasetPenRole: {
     // default to the color set for the brush (or it’s defaults)
     // but only if no per model override was set
     if ( !modelData( role ).isValid() ) {
         QBrush brush = qVariantValue<QBrush>( headerData( section, orientation, DatasetBrushRole ) )
         return QPen( brush.color() );
     }
 }
 default:
     break;
 }

 return QVariant();
}
```

7.13.3.9 const QMap< int, QMap< int, QVariant > > AttributesModel::horizontalHeaderDataMap () const `[protected]`

needed for serialization

Definition at line 526 of file KDChartAttributesModel.cpp.

```
{
  return mHorizontalHeaderDataMap;
}
```

7.13.3.10 QModelIndex KDChart::AbstractProxyModel::index (int *row*, int *col*, const QModelIndex & *index*) const `[inherited]`

Definition at line 53 of file KDChartAbstractProxyModel.cpp.

References KDChart::AbstractProxyModel::mapFromSource(), and KDChart::AbstractProxyModel::mapToSource().

Referenced by setHeaderData(), and setModelData().

```
{
    Q_ASSERT(sourceModel());
    return mapFromSource(sourceModel()->index( row, col, mapToSource(index) ));
}
```

7.13.3.11 void AttributesModel::initFrom (const AttributesModel ∗ *other*)

Definition at line 67 of file KDChartAttributesModel.cpp.

References mDataMap, mHorizontalHeaderDataMap, mModelDataMap, mVerticalHeaderDataMap, paletteType(), and setPaletteType().

Referenced by KDChart::AbstractDiagram::setModel().

```
{
    if( other == this || ! other ) return;

    mDataMap = other->mDataMap;
    mHorizontalHeaderDataMap = other->mHorizontalHeaderDataMap;
    mVerticalHeaderDataMap = other->mVerticalHeaderDataMap;
    mModelDataMap = other->mModelDataMap;

    setPaletteType( other->paletteType() );
}
```

7.13.3.12 bool AttributesModel::isKnownAttributesRole (int *role*) const

Returns whether the given role corresponds to one of the known internally used ones.

Definition at line 396 of file KDChartAttributesModel.cpp.

References KDChart::BarAttributesRole, KDChart::DataHiddenRole, KDChart::DatasetBrushRole, KDChart::DatasetPenRole, KDChart::DataValueLabelAttributesRole, KDChart::LineAttributesRole, KDChart::PieAttributesRole, KDChart::ThreeDAttributesRole, KDChart::ThreeDBarAttributesRole, KDChart::ThreeDLineAttributesRole, and KDChart::ThreeDPieAttributesRole.

Referenced by data(), setData(), and setHeaderData().

```
{
    bool oneOfOurs = false;
    switch( role ) {
      // fallthrough intended
      case DataValueLabelAttributesRole:
      case DatasetBrushRole:
      case DatasetPenRole:
      case ThreeDAttributesRole:
      case LineAttributesRole:
      case ThreeDLineAttributesRole:
      case BarAttributesRole:
```

```
      case ThreeDBarAttributesRole:
      case PieAttributesRole:
      case ThreeDPieAttributesRole:
      case DataHiddenRole:
            oneOfOurs = true;
        default:
        break;
    }
    return oneOfOurs;
}
```

7.13.3.13 QModelIndex KDChart::AbstractProxyModel::mapFromSource (const QModelIndex & *sourceIndex*) const `[inherited]`

Definition at line 23 of file KDChartAbstractProxyModel.cpp.

Referenced by KDChart::AbstractProxyModel::index(), and KDChart::AbstractProxyModel::parent().

```
{
  if ( !sourceIndex.isValid() )
    return QModelIndex();
  //qDebug() << "sourceIndex.model()="<<sourceIndex.model();
  //qDebug() << "model()="<<sourceModel();
  Q_ASSERT( sourceIndex.model() == sourceModel() );

  // Create an index that preserves the internal pointer from the source;
  // this way AbstractProxyModel preserves the structure of the source model
  return createIndex( sourceIndex.row(), sourceIndex.column(), sourceIndex.internalPointer() );
}
```

7.13.3.14 QModelIndex KDChart::AbstractProxyModel::mapToSource (const QModelIndex & *proxyIndex*) const `[inherited]`

Definition at line 36 of file KDChartAbstractProxyModel.cpp.

Referenced by columnCount(), data(), KDChart::AbstractProxyModel::index(), KDChart::AbstractProxyModel::parent(), rowCount(), and setData().

```
{
  if ( !proxyIndex.isValid() )
    return QModelIndex();
  Q_ASSERT( proxyIndex.model() == this );
  // So here we need to create a source index which holds that internal pointer.
  // No way to pass it to sourceModel()->index... so we have to do the ugly way:
  QModelIndex sourceIndex;
  KDPrivateModelIndex* hack = reinterpret_cast<KDPrivateModelIndex*>(&sourceIndex);
  hack->r = proxyIndex.row();
  hack->c = proxyIndex.column();
  hack->p = proxyIndex.internalPointer();
  hack->m = sourceModel();
  Q_ASSERT( sourceIndex.isValid() );
  return sourceIndex;
}
```

7.13.3.15 QVariant KDChart::AttributesModel::modelData (int *role*) const

Definition at line 492 of file KDChartAttributesModel.cpp.

Referenced by data(), and headerData().

```
{
    return mModelDataMap.value( role, QVariant() );
}
```

7.13.3.16 const QMap< int, QVariant > AttributesModel::modelDataMap () const `[protected]`

needed for serialization

Definition at line 536 of file KDChartAttributesModel.cpp.

```
{
    return mModelDataMap;
}
```

7.13.3.17 AttributesModel::PaletteType AttributesModel::paletteType () const

Definition at line 478 of file KDChartAttributesModel.cpp.

Referenced by headerData(), and initFrom().

```
{
    return mPaletteType;
}
```

7.13.3.18 QModelIndex KDChart::AbstractProxyModel::parent (const QModelIndex & *index*) const `[inherited]`

Definition at line 59 of file KDChartAbstractProxyModel.cpp.

References KDChart::AbstractProxyModel::mapFromSource(), and KDChart::AbstractProxyModel::mapToSource().

```
{
    Q_ASSERT(sourceModel());
    return mapFromSource(sourceModel()->parent( mapToSource(index) ));
}
```

7.13.3.19 bool AttributesModel::resetData (const QModelIndex & *index*, int *role* = Qt::DisplayRole)

Remove any explicit attributes settings that might have been specified before.

Definition at line 447 of file KDChartAttributesModel.cpp.

References setData().

```
{
    return setData ( index, QVariant(), role );
}
```

7.13.3.20 bool AttributesModel::resetHeaderData (int *section*, Qt::Orientation *orientation*, int *role* = Qt::DisplayRole)

Remove any explicit attributes settings that might have been specified before.

Definition at line 468 of file KDChartAttributesModel.cpp.

References setHeaderData().

```
469 {
470     return setHeaderData ( section, orientation, QVariant(), role );
471 }
```

7.13.3.21 int AttributesModel::rowCount (const QModelIndex &) const

[reimplemented]

Definition at line 497 of file KDChartAttributesModel.cpp.

References KDChart::AbstractProxyModel::mapToSource().

Referenced by setHeaderData(), and setModelData().

```
498 {
499     Q_ASSERT(sourceModel());
500     return sourceModel()->rowCount( mapToSource(index) );
501 }
```

7.13.3.22 bool AttributesModel::setData (const QModelIndex & *index*, const QVariant & *value*, int *role* = Qt::DisplayRole)

[reimplemented]

Definition at line 433 of file KDChartAttributesModel.cpp.

References dataMap(), isKnownAttributesRole(), and KDChart::AbstractProxyModel::mapToSource().

Referenced by resetData(), and KDChart::BarDiagram::setBarAttributes().

```
434 {
435     if ( !isKnownAttributesRole( role ) ) {
436         return sourceModel()->setData( mapToSource(index), value, role );
437     } else {
438         QMap< int,  QMap< int, QVariant> > &colDataMap = mDataMap[ index.column() ];
439         QMap<int, QVariant> &dataMap = colDataMap[ index.row() ];
440         //qDebug() <<  "AttributesModel::setData" <<"role" << role << "value" << value;
441         dataMap.insert( role, value );
442         emit attributesChanged( index, index );
443         return true;
444     }
445 }
```

7.13.3.23 void AttributesModel::setDataMap (const QMap< int, QMap< int, QMap< int, QVariant > > > *map*) `[protected]`

needed for serialization

Definition at line 542 of file KDChartAttributesModel.cpp.

```
{
    mDataMap = map;
}
```

7.13.3.24 bool AttributesModel::setHeaderData (int *section*, Qt::Orientation *orientation*, const QVariant & *value*, int *role* = Qt::DisplayRole)

[reimplemented]

Definition at line 452 of file KDChartAttributesModel.cpp.

References dataMap(), KDChart::AbstractProxyModel::index(), isKnownAttributesRole(), and rowCount().

Referenced by resetHeaderData().

```
{
    if ( !isKnownAttributesRole( role ) ) {
        return sourceModel()->setHeaderData( section, orientation, value, role );
    } else {
        QMap<int,  QMap<int, QVariant> > &sectionDataMap
            = orientation == Qt::Horizontal ? mHorizontalHeaderDataMap : mVerticalHeaderDataMap;
        QMap<int, QVariant> &dataMap = sectionDataMap[ section ];
        dataMap.insert( role, value );
        emit attributesChanged( index( 0, section, QModelIndex() ),
                                index( rowCount( QModelIndex() ), section, QModelIndex() ) );
        return true;
    }
}
```

7.13.3.25 void AttributesModel::setHorizontalHeaderDataMap (const QMap< int, QMap< int, QVariant > > *map*) `[protected]`

needed for serialization

Definition at line 547 of file KDChartAttributesModel.cpp.

```
{
    mHorizontalHeaderDataMap = map;
}
```

7.13.3.26 bool KDChart::AttributesModel::setModelData (const QVariant *value*, int *role*)

Definition at line 483 of file KDChartAttributesModel.cpp.

References columnCount(), KDChart::AbstractProxyModel::index(), and rowCount().

```
{
    mModelDataMap.insert( role, value );
    emit attributesChanged( index( 0, 0, QModelIndex() ),
                            index( rowCount( QModelIndex() ),
                                    columnCount( QModelIndex() ), QModelIndex() ) );
    return true;
}
```

7.13.3.27 void AttributesModel::setModelDataMap (const QMap< int, QVariant > *map*) `[protected]`

needed for serialization

Definition at line 557 of file KDChartAttributesModel.cpp.

```
558 {
559     mModelDataMap = map;
560 }
```

7.13.3.28 void AttributesModel::setPaletteType (PaletteType *type*)

Sets the palettetype used by this attributesmodel.

Definition at line 473 of file KDChartAttributesModel.cpp.

Referenced by initFrom().

```
474 {
475     mPaletteType = type;
476 }
```

7.13.3.29 void AttributesModel::setSourceModel (QAbstractItemModel ∗ *sourceModel*)

[reimplemented]

Definition at line 509 of file KDChartAttributesModel.cpp.

Referenced by AttributesModel().

```
510 {
511     if( this->sourceModel() != 0 )
512         disconnect( this->sourceModel(), SIGNAL( dataChanged( const QModelIndex&, const QModelIndex&))
513                                    this, SIGNAL( dataChanged( const QModelIndex&, const QModelIndex&)))
514     QAbstractProxyModel::setSourceModel( sourceModel );
515     if( this->sourceModel() != NULL )
516         connect( this->sourceModel(), SIGNAL( dataChanged( const QModelIndex&, const QModelIndex&)),
517                                  this, SIGNAL( dataChanged( const QModelIndex&, const QModelIndex&)));
518 }
```

7.13.3.30 void AttributesModel::setVerticalHeaderDataMap (const QMap< int, QMap< int, QVariant > > *map*) `[protected]`

needed for serialization

Definition at line 552 of file KDChartAttributesModel.cpp.

```
553 {
554     mVerticalHeaderDataMap = map;
555 }
```

7.13.3.31 const QMap< int, QMap< int, QVariant > > AttributesModel::verticalHeaderDataMap () const `[protected]`

needed for serialization

Definition at line 531 of file KDChartAttributesModel.cpp.

```
{
    return mVerticalHeaderDataMap;
}
```

7.13.4 Member Data Documentation

7.13.4.1 Q_SIGNALS KDChart::AttributesModel::__pad0__

Definition at line 125 of file KDChartAttributesModel.h.

The documentation for this class was generated from the following files:

- KDChartAttributesModel.h
- KDChartAttributesModel.cpp

7.14 KDChart::AutoSpacerLayoutItem Class Reference

`#include <KDChartLayoutItems.h>`

Inheritance diagram for KDChart::AutoSpacerLayoutItem:Collaboration diagram for KDChart::AutoSpacerLayoutItem:

Public Member Functions

- AutoSpacerLayoutItem (bool layoutIsAtTopPosition, QHBoxLayout ∗rightLeftLayout, bool layoutIsAtLeftPosition, QVBoxLayout ∗topBottomLayout)
- virtual Qt::Orientations expandingDirections () const
- virtual QRect geometry () const
- virtual bool isEmpty () const
- virtual QSize maximumSize () const
- virtual QSize minimumSize () const
- virtual void paint (QPainter ∗)
- virtual void paintAll (QPainter &painter)

  Default impl: just call paint.

- virtual void paintCtx (PaintContext ∗context)

  Default impl: Paint the complete item using its layouted position and size.

- QLayout ∗ parentLayout ()
- void removeFromParentLayout ()
- virtual void setGeometry (const QRect &r)
- void setParentLayout (QLayout ∗lay)
- virtual void setParentWidget (QWidget ∗widget)

  Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

- virtual QSize sizeHint () const
- virtual void sizeHintChanged () const

  Report changed size hint: ask the parent widget to recalculate the layout.

Protected Attributes

- QWidget ∗ mParent
- QLayout ∗ mParentLayout

7.14.1 Constructor & Destructor Documentation

7.14.1.1 KDChart::AutoSpacerLayoutItem::AutoSpacerLayoutItem (bool *layoutIsAtTopPosition*, QHBoxLayout ∗ *rightLeftLayout*, bool *layoutIsAtLeftPosition*, QVBoxLayout ∗ *topBottomLayout*)

Definition at line 756 of file KDChartLayoutItems.cpp.

```
759     : AbstractLayoutItem( Qt::AlignCenter )
760     , mLayoutIsAtTopPosition(  layoutIsAtTopPosition )
761     , mRightLeftLayout( rightLeftLayout )
762     , mLayoutIsAtLeftPosition( layoutIsAtLeftPosition )
763     , mTopBottomLayout( topBottomLayout )
764 {
765 }
```

7.14.2 Member Function Documentation

7.14.2.1 Qt::Orientations KDChart::AutoSpacerLayoutItem::expandingDirections () const `[virtual]`

Definition at line 767 of file KDChartLayoutItems.cpp.

```
768 {
769     return 0; // Grow neither vertically nor horizontally
770 }
```

7.14.2.2 QRect KDChart::AutoSpacerLayoutItem::geometry () const `[virtual]`

Definition at line 772 of file KDChartLayoutItems.cpp.

```
773 {
774     return mRect;
775 }
```

7.14.2.3 bool KDChart::AutoSpacerLayoutItem::isEmpty () const `[virtual]`

Definition at line 777 of file KDChartLayoutItems.cpp.

```
778 {
779     return true; // never empty, otherwise the layout item would not exist
780 }
```

7.14.2.4 QSize KDChart::AutoSpacerLayoutItem::maximumSize () const `[virtual]`

Definition at line 782 of file KDChartLayoutItems.cpp.

References sizeHint().

```
783 {
784     return sizeHint();
785 }
```

7.14.2.5 QSize KDChart::AutoSpacerLayoutItem::minimumSize () const `[virtual]`

Definition at line 787 of file KDChartLayoutItems.cpp.

References sizeHint().

```
788 {
789     return sizeHint();
790 }
```

7.14.2.6 void KDChart::AutoSpacerLayoutItem::paint (QPainter ∗) `[virtual]`

Implements KDChart::AbstractLayoutItem.

Definition at line 861 of file KDChartLayoutItems.cpp.

```
{
    if( mParentLayout && mRect.isValid() && mCachedSize.isValid() &&
        mCommonBrush.style() != Qt::NoBrush )
    {
        QPoint p1( mRect.topLeft() );
        QPoint p2( mRect.bottomRight() );
        if( mLayoutIsAtLeftPosition )
            p1.rx() += mCachedSize.width() - mParentLayout->spacing();
        else
            p2.rx() -= mCachedSize.width() - mParentLayout->spacing();
        if( mLayoutIsAtTopPosition ){
            p1.ry() += mCachedSize.height() - mParentLayout->spacing() - 1;
            p2.ry() -= 1;
        }else
            p2.ry() -= mCachedSize.height() - mParentLayout->spacing() - 1;
        //qDebug() << mLayoutIsAtTopPosition << mLayoutIsAtLeftPosition;
        //qDebug() << mRect;
        //qDebug() << mParentLayout->margin();
        //qDebug() << QRect( p1, p2 );
        const QPoint oldBrushOrigin( painter->brushOrigin() );
        const QBrush oldBrush( painter->brush() );
        const QPen   oldPen(   painter->pen() );
        const QPointF newTopLeft( painter->deviceMatrix().map( p1 ) );
        painter->setBrushOrigin( newTopLeft );
        painter->setBrush( mCommonBrush );
        painter->setPen( Qt::NoPen );
        painter->drawRect( QRect( p1, p2 ) );
        painter->setBrushOrigin( oldBrushOrigin );
        painter->setBrush( oldBrush );
        painter->setPen( oldPen );
    }
    // debug code:
#if 0
    //qDebug() << "KDChart::AutoSpacerLayoutItem::paint()";
    if( !mRect.isValid() )
        return;

    painter->drawRect( mRect );
    painter->drawLine( QPointF( mRect.x(), mRect.top() ),
                       QPointF( mRect.right(), mRect.bottom() ) );
    painter->drawLine( QPointF( mRect.right(), mRect.top() ),
                       QPointF( mRect.x(), mRect.bottom() ) );
#endif
}
```

7.14.2.7 void KDChart::AbstractLayoutItem::paintAll (QPainter & *painter*) `[virtual, inherited]`

Default impl: just call paint.

Derived classes like KDChart::AbstractArea are providing additional action here.

Reimplemented in KDChart::AbstractArea, and KDChart::TextArea.

Definition at line 69 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint().

```
70 {
71     paint( &painter );
72 }
```

7.14.2.8 void KDChart::AbstractLayoutItem::paintCtx (PaintContext ∗ *context*) `[virtual, inherited]`

Default impl: Paint the complete item using its layouted position and size.

Reimplemented in KDChart::CartesianAxis.

Definition at line 77 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint(), and KDChart::PaintContext::painter().

```
78 {
79     if( context )
80         paint( context->painter() );
81 }
```

7.14.2.9 QLayout∗ KDChart::AbstractLayoutItem::parentLayout () `[inherited]`

Definition at line 74 of file KDChartLayoutItems.h.

```
75         {
76             return mParentLayout;
77         }
```

7.14.2.10 void KDChart::AbstractLayoutItem::removeFromParentLayout () `[inherited]`

Definition at line 78 of file KDChartLayoutItems.h.

Referenced by KDChart::Chart::takeCoordinatePlane().

```
79         {
80             if( mParentLayout ){
81                 if( widget() )
82                     mParentLayout->removeWidget( widget() );
83                 else
84                     mParentLayout->removeItem( this );
85             }
86         }
```

7.14.2.11 void KDChart::AutoSpacerLayoutItem::setGeometry (const QRect & *r*) `[virtual]`

Definition at line 792 of file KDChartLayoutItems.cpp.

```
793 {
794     mRect = r;
795 }
```

7.14.2.12 void KDChart::AbstractLayoutItem::setParentLayout (QLayout ∗ *lay*) `[inherited]`

Definition at line 70 of file KDChartLayoutItems.h.

```
        {
            mParentLayout = lay;
        }
```

7.14.2.13 void KDChart::AbstractLayoutItem::setParentWidget (QWidget ∗ *widget*) `[virtual, inherited]`

Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

Thus, you need to call setParentWidget on every item, that has a non-fixed size.

Definition at line 64 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::mParent.

Referenced by KDChart::Legend::buildLegend(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
{
    mParent = widget;
}
```

7.14.2.14 QSize KDChart::AutoSpacerLayoutItem::sizeHint () const `[virtual]`

Definition at line 817 of file KDChartLayoutItems.cpp.

References KDChart::AbstractArea::bottomOverlap(), KDChart::AbstractArea::leftOverlap(), KDChart::AbstractArea::rightOverlap(), KDChart::AbstractArea::topOverlap(), and updateCommonBrush().

Referenced by maximumSize(), and minimumSize().

```
{
    QBrush commonBrush;
    bool bStart=true;
    // calculate the maximal overlap of the top/bottom axes:
    int topBottomOverlap = 0;
    if( mTopBottomLayout ){
        for (int i = 0; i < mTopBottomLayout->count(); ++i){
            AbstractArea* area = dynamic_cast<AbstractArea*>(mTopBottomLayout->itemAt(i));
            if( area ){
                //qDebug() << "AutoSpacerLayoutItem testing" << area;
                topBottomOverlap =
                    mLayoutIsAtLeftPosition
                    ? qMax( topBottomOverlap, area->rightOverlap() )
                    : qMax( topBottomOverlap, area->leftOverlap() );
                updateCommonBrush( commonBrush, bStart, *area );
            }
        }
    }
    // calculate the maximal overlap of the left/right axes:
    int leftRightOverlap = 0;
    if( mRightLeftLayout ){
        for (int i = 0; i < mRightLeftLayout->count(); ++i){
            AbstractArea* area = dynamic_cast<AbstractArea*>(mRightLeftLayout->itemAt(i));
            if( area ){
                //qDebug() << "AutoSpacerLayoutItem testing" << area;
                leftRightOverlap =
                        mLayoutIsAtTopPosition
                        ? qMax( leftRightOverlap, area->bottomOverlap() )
                        : qMax( leftRightOverlap, area->topOverlap() );
                updateCommonBrush( commonBrush, bStart, *area );
            }
        }
    }
    if( topBottomOverlap > 0 && leftRightOverlap > 0 )
        mCommonBrush = commonBrush;
    else
        mCommonBrush = QBrush();
    mCachedSize = QSize( topBottomOverlap, leftRightOverlap );
    //qDebug() << mCachedSize;
    return mCachedSize;
}
```

7.14.2.15 void KDChart::AbstractLayoutItem::sizeHintChanged () const `[virtual, inherited]`

Report changed size hint: ask the parent widget to recalculate the layout.

Definition at line 86 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextLayoutItem::sizeHint().

```
{
    // This is exactly like what QWidget::updateGeometry does.
//  qDebug("KDChart::AbstractLayoutItem::sizeHintChanged() called");
    if( mParent ) {
        if ( mParent->layout() )
            mParent->layout()->invalidate();
        else
            QApplication::postEvent( mParent, new QEvent( QEvent::LayoutRequest ) );
    }
}
```

7.14.3 Member Data Documentation

7.14.3.1 QWidget∗ KDChart::AbstractLayoutItem::mParent `[protected, inherited]`

Definition at line 88 of file KDChartLayoutItems.h.

Referenced by KDChart::AbstractLayoutItem::setParentWidget().

7.14.3.2 QLayout∗ KDChart::AbstractLayoutItem::mParentLayout `[protected, inherited]`

Definition at line 89 of file KDChartLayoutItems.h.

The documentation for this class was generated from the following files:

- KDChartLayoutItems.h
- KDChartLayoutItems.cpp

7.15 KDChart::BackgroundAttributes Class Reference

`#include <KDChartBackgroundAttributes.h>`

Public Types

- enum BackgroundPixmapMode {
  BackgroundPixmapModeNone,
  BackgroundPixmapModeCentered,
  BackgroundPixmapModeScaled,
  BackgroundPixmapModeStretched }

Public Member Functions

- BackgroundAttributes (const BackgroundAttributes &)
- BackgroundAttributes ()
- QBrush brush () const
- bool isEqualTo (const BackgroundAttributes &other, bool ignorePixmap=false) const
- bool isVisible () const
- bool operator!= (const BackgroundAttributes &other) const
- BackgroundAttributes & operator= (const BackgroundAttributes &)
- bool operator== (const BackgroundAttributes &) const
- QPixmap pixmap () const
- BackgroundPixmapMode pixmapMode () const
- void setBrush (const QBrush &brush)
- void setPixmap (const QPixmap &backPixmap)
- void setPixmapMode (BackgroundPixmapMode mode)
- void setVisible (bool visible)
- ~BackgroundAttributes ()

7.15.1 Member Enumeration Documentation

7.15.1.1 enum KDChart::BackgroundAttributes::BackgroundPixmapMode

Enumeration values:

BackgroundPixmapModeNone

BackgroundPixmapModeCentered

BackgroundPixmapModeScaled

BackgroundPixmapModeStretched

Definition at line 49 of file KDChartBackgroundAttributes.h.

```
49                                  { BackgroundPixmapModeNone,
50                                    BackgroundPixmapModeCentered,
51                                    BackgroundPixmapModeScaled,
52                                    BackgroundPixmapModeStretched };
```

7.15.2 Constructor & Destructor Documentation

7.15.2.1 KDChart::BackgroundAttributes::BackgroundAttributes ()

7.15.2.2 KDChart::BackgroundAttributes::BackgroundAttributes (const BackgroundAttributes &)

7.15.2.3 KDChart::BackgroundAttributes::∼BackgroundAttributes ()

7.15.3 Member Function Documentation

7.15.3.1 QBrush KDChart::BackgroundAttributes::brush () const

Referenced by operator<<(), KDChart::AbstractAreaBase::paintBackgroundAttributes(), and updateCommonBrush().

7.15.3.2 bool KDChart::BackgroundAttributes::isEqualTo (const BackgroundAttributes & *other*, bool *ignorePixmap* = false) const

7.15.3.3 bool KDChart::BackgroundAttributes::isVisible () const

Referenced by operator<<(), KDChart::AbstractAreaBase::paintBackgroundAttributes(), and updateCommonBrush().

7.15.3.4 bool KDChart::BackgroundAttributes::operator!= (const BackgroundAttributes & *other*) const

Definition at line 67 of file KDChartBackgroundAttributes.h.

```
67 { return !operator==(other); }
```

7.15.3.5 BackgroundAttributes& KDChart::BackgroundAttributes::operator= (const BackgroundAttributes &)

7.15.3.6 bool KDChart::BackgroundAttributes::operator== (const BackgroundAttributes &) const

7.15.3.7 QPixmap KDChart::BackgroundAttributes::pixmap () const

Referenced by operator<<(), and KDChart::AbstractAreaBase::paintBackgroundAttributes().

7.15.3.8 BackgroundPixmapMode KDChart::BackgroundAttributes::pixmapMode () const

Referenced by operator<<(), KDChart::AbstractAreaBase::paintBackgroundAttributes(), and updateCommonBrush().

7.15.3.9 void KDChart::BackgroundAttributes::setBrush (const QBrush & *brush*)

7.15.3.10 void KDChart::BackgroundAttributes::setPixmap (const QPixmap & *backPixmap*)

7.15.3.11 void KDChart::BackgroundAttributes::setPixmapMode (BackgroundPixmapMode *mode*)

7.15.3.12 void KDChart::BackgroundAttributes::setVisible (bool *visible*)

The documentation for this class was generated from the following file:

- KDChartBackgroundAttributes.h

7.16 KDChart::BarAttributes Class Reference

`#include <KDChartBarAttributes.h>`

Collaboration diagram for KDChart::BarAttributes:

Public Member Functions

- BarAttributes (const BarAttributes &)
- BarAttributes ()
- qreal barGapFactor () const
- bool drawSolidExcessArrows () const
- qreal fixedBarWidth () const
- qreal fixedDataValueGap () const
- qreal fixedValueBlockGap () const
- qreal groupGapFactor () const
- bool operator!= (const BarAttributes &other) const
- BarAttributes & operator= (const BarAttributes &)
- bool operator== (const BarAttributes &) const
- void setBarGapFactor (qreal gapFactor)
- void setDrawSolidExcessArrows (bool solidArrows)
- void setFixedBarWidth (qreal width)
- void setFixedDataValueGap (qreal gap)
- void setFixedValueBlockGap (qreal gap)
- void setGroupGapFactor (qreal gapFactor)
- void setUseFixedBarWidth (bool useFixedBarWidth)
- void setUseFixedDataValueGap (bool gapIsFixed)
- void setUseFixedValueBlockGap (bool gapIsFixed)
- bool useFixedBarWidth () const
- bool useFixedDataValueGap () const
- bool useFixedValueBlockGap () const
- ~BarAttributes ()

7.16.1 Constructor & Destructor Documentation

7.16.1.1 KDChart::BarAttributes::BarAttributes ()

7.16.1.2 KDChart::BarAttributes::BarAttributes (const BarAttributes &)

7.16.1.3 KDChart::BarAttributes::~BarAttributes ()

7.16.2 Member Function Documentation

7.16.2.1 qreal KDChart::BarAttributes::barGapFactor () const

7.16.2.2 bool KDChart::BarAttributes::drawSolidExcessArrows () const

7.16.2.3 qreal KDChart::BarAttributes::fixedBarWidth () const

Referenced by KDChart::BarDiagram::paint().

7.16.2.4 qreal KDChart::BarAttributes::fixedDataValueGap () const

Referenced by KDChart::BarDiagram::paint().

7.16.2.5 qreal KDChart::BarAttributes::fixedValueBlockGap () const

Referenced by KDChart::BarDiagram::paint().

7.16.2.6 qreal KDChart::BarAttributes::groupGapFactor () const

7.16.2.7 bool KDChart::BarAttributes::operator!= (const BarAttributes & *other*) const

Definition at line 71 of file KDChartBarAttributes.h.

```
71 { return !operator==(other); }
```

7.16.2.8 BarAttributes& KDChart::BarAttributes::operator= (const BarAttributes &)

7.16.2.9 bool KDChart::BarAttributes::operator== (const BarAttributes &) const

7.16.2.10 void KDChart::BarAttributes::setBarGapFactor (qreal *gapFactor*)

7.16.2.11 void KDChart::BarAttributes::setDrawSolidExcessArrows (bool *solidArrows*)

7.16.2.12 void KDChart::BarAttributes::setFixedBarWidth (qreal *width*)

7.16.2.13 void KDChart::BarAttributes::setFixedDataValueGap (qreal *gap*)

7.16.2.14 void KDChart::BarAttributes::setFixedValueBlockGap (qreal *gap*)

7.16.2.15 void KDChart::BarAttributes::setGroupGapFactor (qreal *gapFactor*)

7.16.2.16 void KDChart::BarAttributes::setUseFixedBarWidth (bool *useFixedBarWidth*)

7.16.2.17 void KDChart::BarAttributes::setUseFixedDataValueGap (bool *gapIsFixed*)

7.16.2.18 void KDChart::BarAttributes::setUseFixedValueBlockGap (bool *gapIsFixed*)

7.16.2.19 bool KDChart::BarAttributes::useFixedBarWidth () const

Referenced by KDChart::BarDiagram::paint().

7.16.2.20 bool KDChart::BarAttributes::useFixedDataValueGap () const

Referenced by KDChart::BarDiagram::paint().

7.16.2.21 bool KDChart::BarAttributes::useFixedValueBlockGap () const

Referenced by KDChart::BarDiagram::paint().

The documentation for this class was generated from the following file:

- KDChartBarAttributes.h

7.17 KDChart::BarDiagram Class Reference

`#include <KDChartBarDiagram.h>`

Inheritance diagram for KDChart::BarDiagram:Collaboration diagram for KDChart::BarDiagram:

Public Types

- enum BarType {
  Normal,
  Stacked,
  Percent,
  Rows }

Public Member Functions

- virtual void addAxis (CartesianAxis ∗axis)

  Add the axis to the diagram.

- bool allowOverlappingDataValueTexts () const
- bool antiAliasing () const
- virtual AttributesModel ∗ attributesModel () const

  Returns the AttributesModel, that is used by this diagram.

- virtual KDChart::CartesianAxisList axes () const
- BarAttributes barAttributes (const QModelIndex &index) const
- BarAttributes barAttributes (int column) const
- BarAttributes barAttributes () const
- BarDiagram (QWidget ∗parent=0, CartesianCoordinatePlane ∗plane=0)
- QBrush brush (const QModelIndex &index) const

  Retrieve the brush to be used, for painting the datapoint at the given index in the model.

- QBrush brush (int dataset) const

  Retrieve the brush to be used for the given dataset.

- QBrush brush () const

  Retrieve the brush to be used for painting datapoints globally.

- virtual BarDiagram ∗ clone () const
- bool compare (const AbstractDiagram ∗other) const

  Returns true if both diagrams have the same settings.

- bool compare (const AbstractCartesianDiagram ∗other) const

  Returns true if both diagrams have the same settings.

- AbstractCoordinatePlane ∗ coordinatePlane () const

  The coordinate plane associated with the diagram.

- const QPair< QPointF, QPointF > dataBoundaries () const

Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

- virtual void dataChanged (const QModelIndex &topLeft, const QModelIndex &bottomRight)

  [reimplemented]

- QList< QBrush > datasetBrushes () const

  The set of dataset brushes currently used, for use in legends, etc.

- int datasetDimension () const

  The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

- QStringList datasetLabels () const

  The set of dataset labels currently displayed, for use in legends, etc.

- QList< MarkerAttributes > datasetMarkers () const

  The set of dataset markers currently used, for use in legends, etc.

- QList< QPen > datasetPens () const

  The set of dataset pens currently used, for use in legends, etc.

- DataValueAttributes dataValueAttributes (const QModelIndex &index) const

  Retrieve the DataValueAttributes for the given index.

- DataValueAttributes dataValueAttributes (int column) const

  Retrieve the DataValueAttributes for the given dataset.

- DataValueAttributes dataValueAttributes () const

  Retrieve the DataValueAttributes speficied globally.

- virtual void doItemsLayout ()

  [reimplemented]

- virtual int horizontalOffset () const

  [reimplemented]

- virtual QModelIndex indexAt (const QPoint &point) const

  [reimplemented]

- bool isHidden (const QModelIndex &index) const

  Retrieve the hidden status for the given index.

- bool isHidden (int column) const

  Retrieve the hidden status for the given dataset.

- bool isHidden () const

  Retrieve the hidden status speficied globally.

- virtual bool isIndexHidden (const QModelIndex &index) const

[reimplemented]

- QStringList itemRowLabels () const

  The set of item row labels currently displayed, for use in Abscissa axes, etc.

- virtual void layoutPlanes ()
- virtual QModelIndex moveCursor (CursorAction cursorAction, Qt::KeyboardModifiers modifiers)

  [reimplemented]

- const int numberOfAbscissaSegments () const

  [reimplemented]

- const int numberOfOrdinateSegments () const

  [reimplemented]

- void paintDataValueText (QPainter ∗painter, const QModelIndex &index, const QPointF &pos, double value)
- void paintMarker (QPainter ∗painter, const QModelIndex &index, const QPointF &pos)
- virtual void paintMarker (QPainter ∗painter, const MarkerAttributes &markerAttributes, const QBrush &brush, const QPen &, const QPointF &point, const QSizeF &size)
- QPen pen (const QModelIndex &index) const

  Retrieve the pen to be used, for painting the datapoint at the given index in the model.

- QPen pen (int dataset) const

  Retrieve the pen to be used for the given dataset.

- QPen pen () const

  Retrieve the pen to be used for painting datapoints globally.

- bool percentMode () const
- virtual AbstractCartesianDiagram ∗ referenceDiagram () const
- virtual QPointF referenceDiagramOffset () const
- void resize (const QSizeF &area)

  Called by the widget's sizeEvent.

- virtual void scrollTo (const QModelIndex &index, ScrollHint hint=EnsureVisible)

  [reimplemented]

- void setAllowOverlappingDataValueTexts (bool allow)

  Set whether data value labels are allowed to overlap.

- void setAntiAliasing (bool enabled)

  Set whether anti-aliasing is to be used while rendering this diagram.

- virtual void setAttributesModel (AttributesModel ∗model)

  Associate an AttributesModel with this diagram.

- void setBarAttributes (const QModelIndex &index, const BarAttributes &a)
- void setBarAttributes (int column, const BarAttributes &a)
- void setBarAttributes (const BarAttributes &a)

- void setBrush (const QBrush &brush)

  Set the brush to be used, for painting all datasets in the model.

- void setBrush (int dataset, const QBrush &brush)

  Set the brush to be used, for painting the given dataset.

- void setBrush (const QModelIndex &index, const QBrush &brush)

  Set the brush to be used, for painting the datapoint at the given index.

- virtual void setCoordinatePlane (AbstractCoordinatePlane ∗plane)

  Set the coordinate plane associated with the diagram.

- void setDatasetDimension (int dimension)

  Sets the dataset dimension of the diagram.

- void setDataValueAttributes (const DataValueAttributes &a)

  Set the DataValueAttributes for all datapoints in the model.

- void setDataValueAttributes (int dataset, const DataValueAttributes &a)

  Set the DataValueAttributes for the given dataset.

- void setDataValueAttributes (const QModelIndex &index, const DataValueAttributes &a)

  Set the DataValueAttributes for the given index.

- void setHidden (bool hidden)

  Hide (or unhide, resp.) all datapoints in the model.

- void setHidden (int column, bool hidden)

  Hide (or unhide, resp.) a dataset.

- void setHidden (const QModelIndex &index, bool hidden)

  Hide (or unhide, resp.) a data cell.

- virtual void setModel (QAbstractItemModel ∗model)

  Associate a model with the diagram.

- void setPen (const QPen &pen)

  Set the pen to be used, for painting all datasets in the model.

- void setPen (int dataset, const QPen &pen)

  Set the pen to be used, for painting the given dataset.

- void setPen (const QModelIndex &index, const QPen &pen)

  Set the pen to be used, for painting the datapoint at the given index.

- void setPercentMode (bool percent)
- virtual void setReferenceDiagram (AbstractCartesianDiagram ∗diagram, const QPointF &offset=QPointF())
- virtual void setRootIndex (const QModelIndex &idx)

  Set the root index in the model, where the diagram starts referencing data for display.

- virtual void setSelection (const QRect &rect, QItemSelectionModel::SelectionFlags command)

  [reimplemented]

- void setThreeDBarAttributes (const QModelIndex &index, const ThreeDBarAttributes &a)
- void setThreeDBarAttributes (int column, const ThreeDBarAttributes &a)
- void setThreeDBarAttributes (const ThreeDBarAttributes &a)
- void setType (BarType type)
- virtual void takeAxis (CartesianAxis $*$axis)

  Removes the axis from the diagram, without deleting it.

- ThreeDBarAttributes threeDBarAttributes (const QModelIndex &index) const
- ThreeDBarAttributes threeDBarAttributes (int column) const
- ThreeDBarAttributes threeDBarAttributes () const
- BarType type () const
- void update () const
- void useDefaultColors ()

  Set the palette to be used, for painting datasets to the default palette.

- void useRainbowColors ()

  Set the palette to be used, for painting datasets to the rainbow palette.

- virtual bool usesExternalAttributesModel () const

  Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

- void useSubduedColors ()

  Set the palette to be used, for painting datasets to the subdued palette.

- virtual int verticalOffset () const

  [reimplemented]

- virtual QRect visualRect (const QModelIndex &index) const

  [reimplemented]

- virtual QRegion visualRegionForSelection (const QItemSelection &selection) const

  [reimplemented]

- virtual ∼BarDiagram ()

Protected Member Functions

- QModelIndex attributesModelRootIndex () const
- const QPair< QPointF, QPointF > calculateDataBoundaries () const

  [reimplemented]

- virtual bool checkInvariants (bool justReturnTheStatus=false) const
- QModelIndex columnToIndex (int column) const
- void dataHidden ()

  This signal is emitted, when the hidden status of at least one data cell was (un)set.

- void modelsChanged ()

  This signal is emitted, when either the model or the AttributesModel is replaced.

- void paint (PaintContext ∗paintContext)

  Draw the diagram contents to the rectangle and painter, that are passed in as part of the paint context.

- virtual void paintDataValueTexts (QPainter ∗painter)
- virtual void paintMarkers (QPainter ∗painter)
- void propertiesChanged ()

  Emitted upon change of a property of the Diagram.

- void resizeEvent (QResizeEvent ∗)
- void setAttributesModelRootIndex (const QModelIndex &)
- void setDataBoundariesDirty () const
- virtual double threeDItemDepth (int column) const
- virtual double threeDItemDepth (const QModelIndex &index) const
- double valueForCell (int row, int column) const

  Helper method, retrieving the data value (DisplayRole) for a given row and column.

Protected Attributes

- Q_SIGNALS __pad0__: void layoutChanged(AbstractDiagram∗)

7.17.1 Member Enumeration Documentation

7.17.1.1 enum KDChart::BarDiagram::BarType

Enumeration values:

Normal

Stacked

Percent

Rows

Definition at line 55 of file KDChartBarDiagram.h.

```
55                  { Normal,
56                    Stacked,
57                    Percent,
58                    Rows };
```

7.17.2 Constructor & Destructor Documentation

7.17.2.1 BarDiagram::BarDiagram (QWidget ∗ *parent* = 0, CartesianCoordinatePlane ∗ *plane* = 0) `[explicit]`

Definition at line 52 of file KDChartBarDiagram.cpp.

Referenced by clone().

```
52                                                                       :
53     AbstractCartesianDiagram( new Private(), parent, plane )
54 {
55     init();
56 }
```

7.17.2.2 BarDiagram::∼BarDiagram () [virtual]

Definition at line 62 of file KDChartBarDiagram.cpp.

```
63 {
64 }
```

7.17.3 Member Function Documentation

7.17.3.1 void AbstractCartesianDiagram::addAxis (CartesianAxis ∗ *axis*) [virtual, inherited]

Add the axis to the diagram.

The diagram takes ownership of the axis and will delete it.

To gain back ownership (e.g. for assigning the axis to another diagram) use the takeAxis method, before calling addAxis on the other diagram.

See also:
takeAxis

Definition at line 89 of file KDChartAbstractCartesianDiagram.cpp.

References KDChart::AbstractAxis::createObserver(), d, and KDChart::AbstractCartesianDiagram::layoutPlanes().

```
90 {
91     if ( !d->axesList.contains( axis ) ) {
92         d->axesList.append( axis );
93         axis->createObserver( this );
94         layoutPlanes();
95     }
96 }
```

7.17.3.2 bool AbstractDiagram::allowOverlappingDataValueTexts () const [inherited]

Returns:
Whether data value labels are allowed to overlap.

Definition at line 446 of file KDChartAbstractDiagram.cpp.

References d.

```
450 {
```

7.17.3.3 bool AbstractDiagram::antiAliasing () const `[inherited]`

Returns:

Whether anti-aliasing is to be used for rendering this diagram.

Definition at line 457 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::paint().

```
461 {
```

7.17.3.4 AttributesModel ∗ AbstractDiagram::attributesModel () const `[virtual, inherited]`

Returns the AttributesModel, that is used by this diagram.

By default each diagram owns its own AttributesModel, which should never be deleted. Only if a user-supplied AttributesModel has been set does the pointer returned here not belong to the diagram.

Returns:

The AttributesModel associated with the diagram.

See also:

setAttributesModel

Definition at line 286 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::RingDiagram::paint(), KDChart::PolarDiagram::paint(), and setBarAttributes().

```
287 {
288     return d->attributesModel;
289 }
```

7.17.3.5 QModelIndex AbstractDiagram::attributesModelRootIndex () const `[protected, inherited]`

returns a QModelIndex pointing into the AttributesModel that corresponds to the root index of the diagram.

Definition at line 310 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::calculateDataBoundaries(), calculateDataBoundaries(), KDChart::LineDiagram::numberOfAbscissaSegments(), numberOfAbscissaSegments(), KDChart::LineDiagram::numberOfOrdinateSegments(), numberOfOrdinateSegments(), KDChart::LineDiagram::paint(), paint(), and KDChart::AbstractDiagram::valueForCell().

```
316 {
```

7.17.3.6 KDChart::CartesianAxisList AbstractCartesianDiagram::axes () const `[virtual, inherited]`

Definition at line 108 of file KDChartAbstractCartesianDiagram.cpp.

References KDChart::CartesianAxisList, and d.

```
{
    return d->axesList;
}
```

7.17.3.7 BarAttributes BarDiagram::barAttributes (const QModelIndex & *index*) const

Definition at line 122 of file KDChartBarDiagram.cpp.

References d.

```
{
    return qVariantValue<BarAttributes>(
        d->attributesModel->data(
            d->attributesModel->mapFromSource( index ),
            KDChart::BarAttributesRole ) );
}
```

7.17.3.8 BarAttributes BarDiagram::barAttributes (int *column*) const

Definition at line 114 of file KDChartBarDiagram.cpp.

References d.

```
{
    return qVariantValue<BarAttributes>(
        d->attributesModel->data(
            d->attributesModel->mapFromSource( columnToIndex( column ) ),
            KDChart::BarAttributesRole ) );
}
```

7.17.3.9 BarAttributes BarDiagram::barAttributes () const

Definition at line 108 of file KDChartBarDiagram.cpp.

References d.

Referenced by paint().

```
{
    return qVariantValue<BarAttributes>(
        d->attributesModel->data( KDChart::BarAttributesRole ) );
}
```

7.17.3.10 QBrush AbstractDiagram::brush (const QModelIndex & *index*) const `[inherited]`

Retrieve the brush to be used, for painting the datapoint at the given index in the model.

Parameters:

index The index of the datapoint in the model.

Returns:

The brush to use for painting.

Definition at line 816 of file KDChartAbstractDiagram.cpp.

```
822                               :
QRect AbstractDiagram::visualRect(const QModelIndex &) const
```

7.17.3.11 QBrush AbstractDiagram::brush (int *dataset*) const `[inherited]`

Retrieve the brush to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the brush for.

Returns:

The brush to use for painting.

Definition at line 808 of file KDChartAbstractDiagram.cpp.

```
815 {
```

7.17.3.12 QBrush AbstractDiagram::brush () const `[inherited]`

Retrieve the brush to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The brush to use for painting.

Definition at line 802 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), and KDChart::AbstractDiagram::paintMarker().

```
807 {
```

7.17.3.13 const QPair< QPointF, QPointF > BarDiagram::calculateDataBoundaries () const `[protected, virtual]`

[reimplemented]

Implements KDChart::AbstractDiagram.

Definition at line 198 of file KDChartBarDiagram.cpp.

References KDChart::AbstractDiagram::attributesModelRootIndex(), KDChart::AbstractDiagram::checkInvariants(), d, and type().

```
{
    if ( !checkInvariants(true) ) return QPair<QPointF, QPointF>( QPointF( 0, 0 ), QPointF( 0, 0 ) );
    const int rowCount = d->attributesModel->rowCount(attributesModelRootIndex());
    const int colCount = d->attributesModel->columnCount(attributesModelRootIndex());

    double xMin = 0;
    double xMax = rowCount;
    double yMin = 0, yMax = 0;
    //double maxThreeDDepth   = 0.0;


    // calculate boundaries for  different line types Normal - Stacked - Percent - Default Normal
    switch ( type() ){
        case BarDiagram::Normal:
        {
            bool bStarting = true;
            for ( int i=0; i<colCount; ++i ) {
                for ( int j=0; j< rowCount; ++j ) {
                    const double value = d->attributesModel->data( d->attributesModel->index( j, i, at
                    // this is always true yMin can be 0 in case all values
                    // are the same
                    // same for yMax it can be zero if all values are negative
                    if( bStarting ){
                        yMin = value;
                        yMax = value;
                        bStarting = false;
                    }else{
                        yMin = qMin( yMin, value );
                        yMax = qMax( yMax, value );
                    }
                }
            }
        }
            break;
        case BarDiagram::Stacked:
        {
            bool bStarting = true;
            for ( int j=0; j< rowCount; ++j ) {
                // calculate sum of values per column - Find out stacked Min/Max
                double stackedValues = 0;
                for ( int i=0; i<colCount ; ++i ) {
                    QModelIndex idx = model()->index( j, i, rootIndex() );
                    stackedValues +=  model()->data( idx ).toDouble();
                    // this is always true yMin can be 0 in case all values
                    // are the same
                    // same for yMax it can be zero if all values are negative
                    if( bStarting ){
                        yMin = stackedValues;
                        yMax = stackedValues;
                        bStarting = false;
                    }else{
                        yMin = qMin( yMin, stackedValues );
                        yMax = qMax( yMax, stackedValues );
                    }
                }
            }
        }
            break;
        case BarDiagram::Percent:
        {
            for ( int i=0; i<colCount; ++i ) {
                for ( int j=0; j< rowCount; ++j ) {
                    // Ordinate should begin at 0 the max value being the 100% pos
                    QModelIndex idx = model()->index( j, i, rootIndex() );
                    // only positive values are handled
                    double value = model()->data( idx ).toDouble();
                    if ( value > 0 )
```

```
                        yMax = qMax( yMax, value );
                }
            }
        }
            break;
        case BarDiagram::Rows:
        {
            qDebug()<< "KDChartBarDiagram::calculateDataBoundaries"
                    << "Sorry Type Rows not implemented yet";
            break;
        }


        default:
             Q_ASSERT_X ( false, "calculateDataBoundaries()",
                         "Type item does not match a defined bar chart Type." );
    }

    // special cases
    if (  yMax == yMin ) {
        if ( yMin == 0.0 )
            yMax = 0.1; //we need at list a range
        else
            yMax = 0.0; // they are the same but negative
    }
    QPointF bottomLeft ( QPointF( xMin, yMin ) );
    QPointF topRight ( QPointF( xMax, yMax ) );

    //qDebug() << "BarDiagram::calculateDataBoundaries () returns ( " << bottomLeft << topRight <<")";
    return QPair<QPointF, QPointF> ( bottomLeft,  topRight );
}
```

7.17.3.14 bool AbstractDiagram::checkInvariants (bool *justReturnTheStatus* = false) const `[protected, virtual, inherited]`

Definition at line 930 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane().

Referenced by KDChart::RingDiagram::calculateDataBoundaries(), KDChart::PolarDiagram::calculateDataBoundaries(), KDChart::PieDiagram::calculateDataBoundaries(), KDChart::LineDiagram::calculateDataBoundaries(), calculateDataBoundaries(), KDChart::RingDiagram::paint(), KDChart::PolarDiagram::paint(), KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), paint(), and KDChart::AbstractDiagram::paintMarker().

```
930                                {
931         Q_ASSERT_X ( model(), "AbstractDiagram::checkInvariants()",
932                     "There is no usable model set, for the diagram." );
933
934         Q_ASSERT_X ( coordinatePlane(), "AbstractDiagram::checkInvariants()",
935                     "There is no usable coordinate plane set, for the diagram." );
936     }
937     return model() && coordinatePlane();
938 }
939
940 int AbstractDiagram::datasetDimension( ) const
```

7.17.3.15 BarDiagram ∗ BarDiagram::clone () const `[virtual]`

Definition at line 66 of file KDChartBarDiagram.cpp.

References BarDiagram(), and d.

```
67 {
68     return new BarDiagram( new Private( *d ) );
69 }
```

7.17.3.16 QModelIndex AbstractDiagram::columnToIndex (int *column*) const `[protected, inherited]`

Definition at line 317 of file KDChartAbstractDiagram.cpp.

```
323 {
```

7.17.3.17 bool AbstractDiagram::compare (const AbstractDiagram ∗ *other*) const `[inherited]`

Returns true if both diagrams have the same settings.

Definition at line 135 of file KDChartAbstractDiagram.cpp.

```
136 {
137     if( other == this ) return true;
138     if( ! other ){
139         //qDebug() << "AbstractDiagram::compare() cannot compare to Null pointer";
140         return false;
141     }
142     /*
143     qDebug() << "\n             AbstractDiagram::compare() QAbstractScrollArea:";
144             // compare QAbstractScrollArea properties
145     qDebug() <<
146             ((horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
147             (verticalScrollBarPolicy()    == other->verticalScrollBarPolicy()));
148     qDebug() << "AbstractDiagram::compare() QFrame:";
149             // compare QFrame properties
150     qDebug() <<
151             ((frameShadow() == other->frameShadow()) &&
152             (frameShape()   == other->frameShape()) &&
153             (frameWidth()   == other->frameWidth()) &&
154             (lineWidth()    == other->lineWidth()) &&
155             (midLineWidth() == other->midLineWidth()));
156     qDebug() << "AbstractDiagram::compare() QAbstractItemView:";
157             // compare QAbstractItemView properties
158     qDebug() <<
159             ((alternatingRowColors() == other->alternatingRowColors()) &&
160             (hasAutoScroll()         == other->hasAutoScroll()) &&
161 #if QT_VERSION > 0x040199
162             (dragDropMode()          == other->dragDropMode()) &&
163             (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
164             (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
165             (verticalScrollMode()    == other->verticalScrollMode()) &&
166 #endif
167             (dragEnabled()           == other->dragEnabled()) &&
168             (editTriggers()          == other->editTriggers()) &&
169             (iconSize()              == other->iconSize()) &&
170             (selectionBehavior()     == other->selectionBehavior()) &&
171             (selectionMode()         == other->selectionMode()) &&
172             (showDropIndicator()     == other->showDropIndicator()) &&
173             (tabKeyNavigation()      == other->tabKeyNavigation()) &&
174             (textElideMode()         == other->textElideMode()));
175     qDebug() << "AbstractDiagram::compare() AttributesModel: ";
176             // compare all of the properties stored in the attributes model
177     qDebug() << attributesModel()->compare( other->attributesModel() );
178     qDebug() << "AbstractDiagram::compare() own:";
179             // compare own properties
180     qDebug() <<
181             ((rootIndex().column()            == other->rootIndex().column()) &&
182             (rootIndex().row()                == other->rootIndex().row()) &&
183             (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
184             (antiAliasing()                   == other->antiAliasing()) &&
185             (percentMode()                    == other->percentMode()) &&
186             (datasetDimension()               == other->datasetDimension()));
187     */
188     return  // compare QAbstractScrollArea properties
189             (horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
190             (verticalScrollBarPolicy()   == other->verticalScrollBarPolicy()) &&
191             // compare QFrame properties
192             (frameShadow()  == other->frameShadow()) &&
193             (frameShape()   == other->frameShape()) &&
194             (frameWidth()   == other->frameWidth()) &&
195             (lineWidth()    == other->lineWidth()) &&
196             (midLineWidth() == other->midLineWidth()) &&
197             // compare QAbstractItemView properties
198             (alternatingRowColors()  == other->alternatingRowColors()) &&
199             (hasAutoScroll()         == other->hasAutoScroll()) &&
200 #if QT_VERSION > 0x040199
201             (dragDropMode()          == other->dragDropMode()) &&
202             (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
203             (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
204             (verticalScrollMode()    == other->verticalScrollMode()) &&
205 #endif
206             (dragEnabled()           == other->dragEnabled()) &&
207             (editTriggers()          == other->editTriggers()) &&
208             (iconSize()              == other->iconSize()) &&
209             (selectionBehavior()     == other->selectionBehavior()) &&
210             (selectionMode()         == other->selectionMode()) &&
211             (showDropIndicator()     == other->showDropIndicator()) &&
212             (tabKeyNavigation()      == other->tabKeyNavigation()) &&
213             (textElideMode()         == other->textElideMode()) &&
214             // compare all of the properties stored in the attributes model
215             attributesModel()->compare( other->attributesModel() ) &&
216             // compare own properties
217             (rootIndex().column()             == other->rootIndex().column()) &&
218             (rootIndex().row()                == other->rootIndex().row()) &&
219             (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
220             (antiAliasing()                   == other->antiAliasing()) &&
221             (percentMode()                    == other->percentMode()) &&
222             (datasetDimension()               == other->datasetDimension());
223 }
```

7.17.3.18 bool AbstractCartesianDiagram::compare (const AbstractCartesianDiagram ∗ *other*) const `[inherited]`

Returns true if both diagrams have the same settings.

Definition at line 52 of file KDChartAbstractCartesianDiagram.cpp.

```
53 {
54     if( other == this ) return true;
55     if( ! other ){
56         //qDebug() << "AbstractCartesianDiagram::compare() cannot compare to Null pointer";
57         return false;
58     }
59     /*
60     qDebug() << "\n             AbstractCartesianDiagram::compare():";
```

```
            // compare own properties
    qDebug() <<
            ((referenceDiagram() == other->referenceDiagram()) &&
            ((! referenceDiagram()) || (referenceDiagramOffset() == other->referenceDiagramOffset()))));
    */
    return  // compare the base class
            ( static_cast<const AbstractDiagram*>(this)->compare( other ) ) &&
            // compare own properties
            (referenceDiagram() == other->referenceDiagram()) &&
            ((! referenceDiagram()) || (referenceDiagramOffset() == other->referenceDiagramOffset()));
}
```

7.17.3.19 AbstractCoordinatePlane ∗ AbstractDiagram::coordinatePlane () const `[inherited]`

The coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. By default this is a CartesianCoordinatePlane.

Returns:

The coordinate plane associated with the diagram.

Definition at line 226 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractCartesianDiagram::layoutPlanes(), KDChart::PolarDiagram::paint(), KDChart::LineDiagram::paint(), paint(), KDChart::AbstractPolarDiagram::polarCoordinatePlane(), and KDChart::AbstractCartesianDiagram::setCoordinatePlane().

```
{
    return d->plane;
}
```

7.17.3.20 const QPair< QPointF, QPointF > AbstractDiagram::dataBoundaries () const `[inherited]`

Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

This method returns a chached result of calculations done by calculateDataBoundaries. Classes derived from AbstractDiagram must implement the calculateDataBoundaries function, to specify their own way of calculating the data boundaries. If derived classes want to force recalculation of the data boundaries, they can call setDataBoundariesDirty()

Returned value is in diagram coordinates.

Definition at line 231 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::calculateDataBoundaries(), and d.

Referenced by KDChart::CartesianCoordinatePlane::getRawDataBoundingRectFromDiagrams(), KDChart::PolarCoordinatePlane::layoutDiagrams(), KDChart::LineDiagram::paint(), and paint().

```
{
    if( d->databoundariesDirty ){
```

```
234         d->databoundaries = calculateDataBoundaries ();
235         d->databoundariesDirty = false;
236     }
237     return d->databoundaries;
238 }
```

7.17.3.21 void AbstractDiagram::dataChanged (const QModelIndex & *topLeft*, const QModelIndex & *bottomRight*) [virtual, inherited]

[reimplemented]

Definition at line 338 of file KDChartAbstractDiagram.cpp.

References d.

```
338 {
339   // We are still too dumb to do intelligent updates...
340   d->databoundariesDirty = true;
341   scheduleDelayedItemsLayout();
342 }
343
344
```

7.17.3.22 void KDChart::AbstractDiagram::dataHidden () [protected, inherited]

This signal is emitted, when the hidden status of at least one data cell was (un)set.

7.17.3.23 QList< QBrush > AbstractDiagram::datasetBrushes () const [inherited]

The set of dataset brushes currently used, for use in legends, etc.

Note:

Cell-level override brushes, if set, take precedence over the dataset values, so you might need to check these too, in order to find the brush, that is used for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 894 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), KDChart::Legend::datasetCount(), and KDChart::Legend::setBrushesFromDiagram().

```
896                                                                                   {
897         QBrush brush = qVariantValue<QBrush>( attributesModel()->headerData( i, Qt::Vertical, DatasetB
898         ret << brush;
899     }
900
901     return ret;
902 }
903
904 QList<QPen> AbstractDiagram::datasetPens() const
```

7.17.3.24 int AbstractDiagram::datasetDimension () const `[inherited]`

The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

For each dimension it will expect one column of values in the model. If the dimensionality is 1, automatic values will be used for the abscissa.

For example a diagram with the default dimension of 1, will have one column per datapoint (the y values) and will use automatic values for the x axis (1, 2, 3, ... n). If the dimension is 2, the diagram will use the first, (and the third, fifth, etc) columns as X values, and the second, (and the fourth, sixth, etc) column as Y values.

Returns:

The dataset dimension of the diagram.

Definition at line 942 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::calculateDataBoundaries(), KDChart::LineDiagram::getCellValues(), KDChart::CartesianCoordinatePlane::getDataDimensionsList(), KDChart::LineDiagram::paint(), and KDChart::LineDiagram::setType().

```
946 {
```

7.17.3.25 QStringList AbstractDiagram::datasetLabels () const `[inherited]`

The set of dataset labels currently displayed, for use in legends, etc.

Returns:

The set of dataset labels currently displayed.

Definition at line 882 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), and KDChart::Legend::datasetCount().

```
883                                                  : " << attributesModel()->columnCount(attributesModel
884     const int columnCount = attributesModel()->columnCount(attributesModelRootIndex());
885     for( int i = datasetDimension()-1; i < columnCount; i += datasetDimension() ){
886         //qDebug() << "dataset label: " << attributesModel()->headerData( i, Qt::Horizontal, Qt::Displ
887         ret << attributesModel()->headerData( i, Qt::Horizontal, Qt::DisplayRole ).toString();
888     }
889     return ret;
890 }
891 
892 QList<QBrush> AbstractDiagram::datasetBrushes() const
```

7.17.3.26 QList< MarkerAttributes > AbstractDiagram::datasetMarkers () const `[inherited]`

The set of dataset markers currently used, for use in legends, etc.

Note:

Cell-level override markers, if set, take precedence over the dataset values, so you might need to check these too, in order to find the marker, that is shown for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 917 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
919                                                                                    {
920         DataValueAttributes a =
921             qVariantValue<DataValueAttributes>( attributesModel()->headerData( i, Qt::Vertical, DataVa
922         const MarkerAttributes &ma = a.markerAttributes();
923         ret << ma;
924     }
925     return ret;
926 }
927
928 bool AbstractDiagram::checkInvariants( bool justReturnTheStatus ) const
```

7.17.3.27 QList< QPen > AbstractDiagram::datasetPens () const `[inherited]`

The set of dataset pens currently used, for use in legends, etc.

Note:

Cell-level override pens, if set, take precedence over the dataset values, so you might need to check these too, in order to find the pens, that is used for a single cell.

Returns:

The current set of dataset pens.

Definition at line 906 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
908                                                                                    {
909         QPen pen = qVariantValue<QPen>( attributesModel()->headerData( i, Qt::Vertical, DatasetPenRole
910         ret << pen;
911     }
912     return ret;
913 }
914
915 QList<MarkerAttributes> AbstractDiagram::datasetMarkers() const
```

7.17.3.28 DataValueAttributes AbstractDiagram::dataValueAttributes (const QModelIndex & *index*) const `[inherited]`

Retrieve the DataValueAttributes for the given index.

This will fall back automatically to what was set at dataset or model level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the attributes for.

Returns:

The DataValueAttributes for the given index.

Definition at line 427 of file KDChartAbstractDiagram.cpp.

```
433 {
```

7.17.3.29 DataValueAttributes AbstractDiagram::dataValueAttributes (int *column*) const `[inherited]`

Retrieve the DataValueAttributes for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the attributes for.

Returns:

The DataValueAttributes for the given dataset.

Definition at line 420 of file KDChartAbstractDiagram.cpp.

```
426 {
```

7.17.3.30 DataValueAttributes AbstractDiagram::dataValueAttributes () const `[inherited]`

Retrieve the DataValueAttributes speficied globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The global DataValueAttributes.

Definition at line 414 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::AbstractDiagram::paintDataValueText(), and KDChart::AbstractDiagram::paintMarker().

```
419 {
```

7.17.3.31 void AbstractDiagram::doItemsLayout () `[virtual, inherited]`

[reimplemented]

Definition at line 329 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::AbstractDiagram::update().

```
329                     {
330         d->plane->layoutDiagrams();
331         update();
332     }
333     QAbstractItemView::doItemsLayout();
334 }
335
336 void AbstractDiagram::dataChanged( const QModelIndex &topLeft,
```

7.17.3.32 int AbstractDiagram::horizontalOffset () const `[virtual, inherited]`

[reimplemented]

Definition at line 839 of file KDChartAbstractDiagram.cpp.

```
841 { return 0; }
```

7.17.3.33 QModelIndex AbstractDiagram::indexAt (const QPoint & *point*) const `[virtual, inherited]`

[reimplemented]

Definition at line 833 of file KDChartAbstractDiagram.cpp.

```
835 { return QModelIndex(); }
```

7.17.3.34 bool AbstractDiagram::isHidden (const QModelIndex & *index*) const `[inherited]`

Retrieve the hidden status for the given index.

This will fall back automatically to what was set at dataset or diagram level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the hidden status for.

Returns:

The hidden status for the given index.

Definition at line 386 of file KDChartAbstractDiagram.cpp.

7.17.3.35 bool AbstractDiagram::isHidden (int *column*) const `[inherited]`

Retrieve the hidden status for the given dataset.

This will fall back automatically to what was set at diagram level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the hidden status for.

Returns:

The hidden status for the given dataset.

Definition at line 379 of file KDChartAbstractDiagram.cpp.

```
385 {
```

7.17.3.36 bool AbstractDiagram::isHidden () const `[inherited]`

Retrieve the hidden status speficied globally.

This will fall back automatically to the default settings (= not hidden), if there are no specific settings.

Returns:
The global hidden status.

Definition at line 373 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), KDChart::LineDiagram::paint(), and KDChart::LineDiagram::valueForCellTesting().

```
{
```

7.17.3.37 bool AbstractDiagram::isIndexHidden (const QModelIndex & *index*) const `[virtual, inherited]`

[reimplemented]

Definition at line 845 of file KDChartAbstractDiagram.cpp.

```
{}
```

7.17.3.38 QStringList AbstractDiagram::itemRowLabels () const `[inherited]`

The set of item row labels currently displayed, for use in Abscissa axes, etc.

Returns:
The set of item row labels currently displayed.

Definition at line 870 of file KDChartAbstractDiagram.cpp.

```
                                                 : " << attributesModel()->rowCount(attributesModelRoc
    const int rowCount = attributesModel()->rowCount(attributesModelRootIndex());
    for( int i = 0; i < rowCount; ++i ){
        //qDebug() << "item row label: " << attributesModel()->headerData( i, Qt::Vertical, Qt::Displa
        ret << attributesModel()->headerData( i, Qt::Vertical, Qt::DisplayRole ).toString();
    }
    return ret;
}

QStringList AbstractDiagram::datasetLabels() const
```

7.17.3.39 void KDChart::AbstractCartesianDiagram::layoutPlanes () `[virtual, inherited]`

Definition at line 113 of file KDChartAbstractCartesianDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane(), and KDChart::AbstractCoordinatePlane::layoutPlanes().

Referenced by KDChart::AbstractCartesianDiagram::addAxis(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
{
    //qDebug() << "KDChart::AbstractCartesianDiagram::layoutPlanes()";
    AbstractCoordinatePlane* plane = coordinatePlane();
    if( plane ){
        plane->layoutPlanes();
        //qDebug() << "KDChart::AbstractCartesianDiagram::layoutPlanes() OK";
    }
}
```

7.17.3.40 void KDChart::AbstractDiagram::modelsChanged () `[protected, inherited]`

This signal is emitted, when either the model or the AttributesModel is replaced.

Referenced by KDChart::AbstractDiagram::setAttributesModel(), and KDChart::AbstractDiagram::setModel().

7.17.3.41 QModelIndex AbstractDiagram::moveCursor (CursorAction *cursorAction*, Qt::KeyboardModifiers *modifiers*) `[virtual, inherited]`

[reimplemented]

Definition at line 836 of file KDChartAbstractDiagram.cpp.

```
{ return 0; }
```

7.17.3.42 const int BarDiagram::numberOfAbscissaSegments () const `[virtual]`

[reimplemented]

Implements KDChart::AbstractCartesianDiagram.

Definition at line 674 of file KDChartBarDiagram.cpp.

References KDChart::AbstractDiagram::attributesModelRootIndex(), and d.

```
{
    return d->attributesModel->rowCount(attributesModelRootIndex());
}
```

7.17.3.43 const int BarDiagram::numberOfOrdinateSegments () const `[virtual]`

[reimplemented]

Implements KDChart::AbstractCartesianDiagram.

Definition at line 679 of file KDChartBarDiagram.cpp.

References KDChart::AbstractDiagram::attributesModelRootIndex(), and d.

```
{
    return d->attributesModel->columnCount(attributesModelRootIndex());
}
```

7.17.3.44 void BarDiagram::paint (PaintContext ∗ *paintContext*) `[protected, virtual]`

Draw the diagram contents to the rectangle and painter, that are passed in as part of the paint context.

Parameters:

paintContext All information needed for painting.

Implements KDChart::AbstractDiagram.

Definition at line 341 of file KDChartBarDiagram.cpp.

References KDChart::AbstractDiagram::attributesModelRootIndex(), barAttributes(), KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractDiagram::coordinatePlane(), d, KDChart::AbstractDiagram::dataBoundaries(), KDChart::AbstractThreeDAttributes::depth(), KDChart::BarAttributes::fixedBarWidth(), KDChart::BarAttributes::fixedDataValueGap(), KDChart::BarAttributes::fixedValueBlockGap(), KDChart::AbstractThreeDAttributes::isEnabled(), KDChart::PaintContext::rectangle(), threeDBarAttributes(), KDChart::CartesianCoordinatePlane::translate(), type(), KDChart::BarAttributes::useFixedBarWidth(), KDChart::BarAttributes::useFixedDataValueGap(), and KDChart::BarAttributes::useFixedValueBlockGap().

```
{
    // note: Not having any data model assigned is no bug
    //       but we can not draw a diagram then either.
    if ( !checkInvariants(true) )
        return;

    // Calculate width
    QPointF boundLeft, boundRight;
    QPair<QPointF,QPointF> boundaries = dataBoundaries();
    if( !AbstractGrid::isBoundariesValid(boundaries) ) return;

    CartesianCoordinatePlane* plane = dynamic_cast<KDChart::CartesianCoordinatePlane*>( coordinatePlan
    if( ! plane ) return;

    boundLeft = plane->translate( boundaries.first );
    boundRight = plane->translate( boundaries.second );
    double width = boundRight.x() - boundLeft.x();
    //calculates and stores the values
    const int rowCount = d->attributesModel->rowCount(attributesModelRootIndex());
    const int colCount = d->attributesModel->columnCount(attributesModelRootIndex());
    DataValueTextInfoList list;
    BarAttributes ba = barAttributes( model()->index( 0, 0, rootIndex() ) );
    double barWidth = 0;
    double maxDepth = 0;
    double spaceBetweenBars = 0;
    double spaceBetweenGroups = 0;
    double groupWidth =  /*ctx->rectangle().width() / ( rowCount + 2 )*/ width/ (rowCount + 2);


    if ( ba.useFixedBarWidth() ) {
        barWidth = ba.fixedBarWidth();
        groupWidth += barWidth;

        // Pending Michel set a min and max value for the groupWidth related to the area.width
        // FixMe
        if ( groupWidth < 0 )
            groupWidth = 0;

        if ( groupWidth  * rowCount > ctx->rectangle().width() )
            groupWidth = ctx->rectangle().width() / rowCount;
    }

    // maxLimit: allow the space between bars to be larger until area.width() is covered by the groups
    double maxLimit = rowCount * (groupWidth + ((colCount-1) * ba.fixedDataValueGap()) );

    //Pending Michel: FixMe
    if ( ba.useFixedDataValueGap() ) {
        if ( ctx->rectangle().width() > maxLimit )
            spaceBetweenBars += ba.fixedDataValueGap();
        else
            spaceBetweenBars = ((ctx->rectangle().width()/rowCount) - groupWidth)/(colCount-1);
   }

    //Pending Michel: FixMe
    if ( ba.useFixedValueBlockGap() )
        spaceBetweenGroups += ba.fixedValueBlockGap();

    calculateValueAndGapWidths( rowCount, colCount,groupWidth,
                                barWidth, spaceBetweenBars, spaceBetweenGroups );

    // paint different bar types: Normal - Stacked - Percent
    switch ( type() )
        {
        case BarDiagram::Normal:
        {
            // we paint the bars for all series next to each other, then move to the next value
            for ( int i=0; i<rowCount; ++i ) {
                double offset = -groupWidth/2 + spaceBetweenGroups/2;
                // case fixed data value gap - handles max and min limits as well
                if ( ba.useFixedDataValueGap() ) {
                    if ( spaceBetweenBars > 0 ) {
                      if ( ctx->rectangle().width() > maxLimit )
                            offset -= ba.fixedDataValueGap();
                      else
                        offset -= ((ctx->rectangle().width()/rowCount) - groupWidth)/(colCount-1);

                    } else {
                        //allow reducing the gap until the bars are displayed next to each other - nul
                        offset += barWidth/2;
                    }
                }

                for ( int j=0; j< colCount; ++j ) {
                    // paint one group
                    const qreal value = d->attributesModel->data( d->attributesModel->index( i, j, att
                    QPointF topPoint = plane->translate( QPointF( i + 0.5, value ) );
                    QPointF bottomPoint = plane->translate( QPointF( i, 0 ) );
                    const double barHeight = bottomPoint.y() - topPoint.y();
                    topPoint.setX( topPoint.x() + offset );

                    const QModelIndex index = model()->index( i, j, rootIndex() );

                    //PENDING Michel: FIXME barWidth
                    const QRectF rect( topPoint, QSizeF( barWidth, barHeight ) );
                    d->appendDataValueTextInfoToList( this, list, index, PositionPoints( rect ),
                            Position::NorthWest, Position::SouthEast,
                            value );
                    paintBars( ctx, index, rect, maxDepth );

                    offset += barWidth + spaceBetweenBars;
                }
            }
        }
            break;
        case BarDiagram::Stacked:
        {
           for ( int i = 0; i<colCount; ++i ) {
               double offset = spaceBetweenGroups;
               for ( int j = 0; j< rowCount; ++j ) {
                   QModelIndex index = model()->index( j, i, rootIndex() );
                   ThreeDBarAttributes threeDAttrs = threeDBarAttributes( index );
                   double value = 0, stackedValues = 0;
                   QPointF point, previousPoint;

                   if ( threeDAttrs.isEnabled() ) {
                       if ( barWidth > 0 )
                           barWidth =  (width - ((offset+(threeDAttrs.depth()))*rowCount))/ rowCount;
                       if ( barWidth <= 0 ) {
                           barWidth = 0;
                           maxDepth = offset - (width/rowCount);
                       }
                   } else
                       barWidth =  (ctx->rectangle().width() - (offset*rowCount))/ rowCount ;

                   value = model()->data( index ).toDouble();
                   for ( int k = i; k >= 0 ; --k )
                       stackedValues += model()->data( model()->index( j, k, rootIndex() ) ).toDouble(
                   point = plane->translate( QPointF( j, stackedValues ) );
                   point.setX( point.x() + offset/2 );
                   previousPoint = plane->translate( QPointF( j, stackedValues - value ) );
                   const double barHeight = previousPoint.y() - point.y();

                   const QRectF rect( point, QSizeF( barWidth , barHeight ) );
                   d->appendDataValueTextInfoToList( this, list, index, PositionPoints( rect ),
                           Position::NorthWest, Position::SouthEast,
                           value );
                   paintBars( ctx, index, rect, maxDepth );
               }

           }
        }
           break;
        case BarDiagram::Percent:
        {
            double maxValue = 100; // always 100 %
            double sumValues = 0;
            QVector <double > sumValuesVector;

            //calculate sum of values for each column and store
            for ( int j=0; j<rowCount; ++j ) {
                for ( int i=0; i<colCount; ++i ) {
                    double tmpValue = model()->data( model()->index( j, i, rootIndex() ) ).toDouble();
                    if ( tmpValue > 0 )
                        sumValues += tmpValue;
                    if ( i == colCount-1 ) {
                        sumValuesVector <<  sumValues ;
                        sumValues = 0;
                    }
                }
            }

            // calculate stacked percent value
            for ( int i = 0; i<colCount; ++i ) {
                double offset = spaceBetweenGroups;
                for ( int j=0; j<rowCount ; ++j ) {
                    double value = 0, stackedValues = 0;
                    QPointF point, previousPoint;
                    QModelIndex index = model()->index( j, i, rootIndex() );
                    ThreeDBarAttributes threeDAttrs = threeDBarAttributes( index );

                    if ( threeDAttrs.isEnabled() ){
                        if ( barWidth > 0 )
                            barWidth =  (width - ((offset+(threeDAttrs.depth()))*rowCount))/ rowCount;
                        if ( barWidth <= 0 ) {
                            barWidth = 0;
                            maxDepth = offset - ( width/rowCount);
                        }
                    }else{
                        barWidth = (ctx->rectangle().width() - (offset*rowCount))/ rowCount;
                    }

                    value = model()->data( index ).toDouble();

                    // calculate stacked percent value
                    // we only take in account positives values for now.
                    for ( int k = i; k >= 0 ; --k ) {
                        double val = model()->data( model()->index( j, k, rootIndex() ) ).toDouble();
                        if ( val > 0)
                            stackedValues += val;
                    }

                    if (  sumValuesVector.at( j ) != 0 && value > 0 ) {
                      point = plane->translate( QPointF( j,  stackedValues/sumValuesVector.at(j)* maxV

                        point.setX( point.x() + offset/2 );

                        previousPoint = plane->translate( QPointF( j, (stackedValues - value)/sumValue
                    }
                    const double barHeight = previousPoint.y() - point.y();

                    const QRectF rect( point, QSizeF( barWidth, barHeight ) );
                    d->appendDataValueTextInfoToList( this, list, index, PositionPoints( rect ),
                            Position::NorthWest, Position::SouthEast,
                            value );
                    paintBars( ctx, index, rect, maxDepth );

                }
            }
        }
        break;
        default:
            Q_ASSERT_X ( false, "paint()",
                         "Type item does not match a defined bar chart Type." );
        }

    // paint all data value texts, but no point markers
    d->paintDataValueTextsAndMarkers( this, ctx, list, false );
}
```

7.17.3.45 void AbstractDiagram::paintDataValueText (QPainter ∗ *painter*, const QModelIndex & *index*, const QPointF & *pos*, double *value*) `[inherited]`

Definition at line 474 of file KDChartAbstractDiagram.cpp.

References KDChart::RelativePosition::alignment(), KDChart::TextAttributes::calculatedFont(), d, KDChart::DataValueAttributes::dataLabel(), KDChart::AbstractDiagram::dataValueAttributes(), KDChart::DataValueAttributes::decimalDigits(), KDChart::TextAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::TextAttributes::pen(), KDChart::DataValueAttributes::position(), KDChart::DataValueAttributes::prefix(), KDChart::TextAttributes::rotation(), KDChart::DataValueAttributes::showRepetitiveDataLabels(), KDChart::DataValueAttributes::suffix(), and KDChart::DataValueAttributes::textAttributes().

Referenced by KDChart::RingDiagram::paint(), and KDChart::PolarDiagram::paint().

```
{
    // paint one data series
    const DataValueAttributes a( dataValueAttributes(index) );
    if ( !a.isVisible() ) return;

    // handle decimal digits
    int decimalDigits = a.decimalDigits();
    int decimalPos = QString::number(  value ).indexOf( QLatin1Char( ’.’ ) );
    QString roundedValue;
    if ( a.dataLabel().isNull() ) {
        if ( decimalPos > 0 && value != 0 )
            roundedValue =  roundValues ( value, decimalPos, decimalDigits );
        else
            roundedValue = QString::number(  value );
    } else
        roundedValue = a.dataLabel();
        // handle prefix and suffix
    if ( !a.prefix().isNull() )
        roundedValue.prepend( a.prefix() );

    if ( !a.suffix().isNull() )
        roundedValue.append( a.suffix() );

    const TextAttributes ta( a.textAttributes() );
    // FIXME draw the non-text bits, background, etc
    if ( ta.isVisible() ) {

        QPointF pt( pos );
        /* for debugging:
        PainterSaver painterSaver( painter );
        painter->setPen( Qt::black );
        painter->drawLine( pos - QPointF( 1,1), pos + QPointF( 1,1) );
        painter->drawLine( pos - QPointF(-1,1), pos + QPointF(-1,1) );
        */

        // adjust the text start point position, if alignment is not Bottom/Left
        const RelativePosition relPos( a.position( value >= 0.0 ) );
        const Qt::Alignment alignBottomLeft = Qt::AlignBottom | Qt::AlignLeft;
        const QFont calculatedFont( ta.calculatedFont( d->plane, KDChartEnums::MeasureOrientationMinim
        //qDebug() << "calculatedFont’s point size:" << calculatedFont.pointSizeF();
        if( (relPos.alignment() & alignBottomLeft) != alignBottomLeft ){
            const QRectF boundRect(
                    d->cachedFontMetrics( calculatedFont, this )->boundingRect( roundedValue ) );
            if( relPos.alignment() & Qt::AlignRight )
                pt.rx() -= boundRect.width();
            else if( relPos.alignment() & Qt::AlignHCenter )
                pt.rx() -= 0.5 * boundRect.width();

            if( relPos.alignment() & Qt::AlignTop )
                pt.ry() += boundRect.height();
            else if( relPos.alignment() & Qt::AlignVCenter )
                pt.ry() += 0.5 * boundRect.height();
        }

        // FIXME draw the non-text bits, background, etc

        if ( a.showRepetitiveDataLabels() ||
             pos.x() <= d->lastX ||
             d->lastRoundedValue != roundedValue ) {
            d->lastRoundedValue = roundedValue;
            d->lastX = pos.x();

            PainterSaver painterSaver( painter );
            painter->setPen( ta.pen() );
            painter->setFont( calculatedFont );
            painter->translate( pt );
            painter->rotate( ta.rotation() );
            painter->drawText( QPointF(0, 0), roundedValue );
        }
    }
}
```

7.17.3.46 void AbstractDiagram::paintDataValueTexts (QPainter ∗ *painter*) `[protected, virtual, inherited]`

Definition at line 576 of file KDChartAbstractDiagram.cpp.

```
                                                                               {
       for ( int j=0; j< rowCount; ++j ) {
           const QModelIndex index = model()->index( j, i, rootIndex() );
           double value = model()->data( index ).toDouble();
           const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
           paintDataValueText( painter, index, pos, value );
       }
    }
}
```

7.17.3.47 void AbstractDiagram::paintMarker (QPainter ∗ *painter*, const QModelIndex & *index*, const QPointF & *pos*) `[inherited]`

Definition at line 592 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::brush(), KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractDiagram::dataValueAttributes(), KDChart::MarkerAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::DataValueAttributes::markerAttributes(), KDChart::MarkerAttributes::markerColor(), KDChart::MarkerAttributes::markerSize(), KDChart::AbstractDiagram::paintMarker(), and KDChart::MarkerAttributes::pen().

```
{

    if ( !checkInvariants() ) return;
    DataValueAttributes a = dataValueAttributes(index);
    if ( !a.isVisible() ) return;
    const MarkerAttributes &ma = a.markerAttributes();
    if ( !ma.isVisible() ) return;

    PainterSaver painterSaver( painter );
    QSizeF maSize( ma.markerSize() );
    QBrush indexBrush( brush( index ) );
    QPen indexPen( ma.pen() );
    if ( ma.markerColor().isValid() )
        indexBrush.setColor( ma.markerColor() );

    paintMarker( painter, ma, indexBrush, indexPen, pos, maSize );
}
```

7.17.3.48 void AbstractDiagram::paintMarker (QPainter ∗ *painter*, const MarkerAttributes & *markerAttributes*, const QBrush & *brush*, const QPen &, const QPointF & *point*, const QSizeF & *size*) `[virtual, inherited]`

Definition at line 614 of file KDChartAbstractDiagram.cpp.

References KDChart::MarkerAttributes::markerStyle().

Referenced by KDChart::MarkerLayoutItem::paintIntoRect(), and KDChart::AbstractDiagram::paintMarker().

```
{

    const QPen oldPen( painter->pen() );
    // Pen is used to paint 4Pixels - 1 Pixel - Ring and FastCross types.
    // make sure to use the brush color - see above in those cases.
    const bool isFourPixels = (markerAttributes.markerStyle() == MarkerAttributes::Marker4Pixels);
    if( isFourPixels || (markerAttributes.markerStyle() == MarkerAttributes::Marker1Pixel) ){
        // for high-performance point charts with tiny point markers:
        painter->setPen( QPen( brush.color().light() ) );
        if( isFourPixels ){
            const qreal x = pos.x();
            const qreal y = pos.y();
            painter->drawLine( QPointF(x-1.0,y-1.0),
                               QPointF(x+1.0,y-1.0) );
            painter->drawLine( QPointF(x-1.0,y),
                               QPointF(x+1.0,y) );
            painter->drawLine( QPointF(x-1.0,y+1.0),
                               QPointF(x+1.0,y+1.0) );
        }
        painter->drawPoint( pos );
    }else{
        PainterSaver painterSaver( painter );
        // we only a solid line surrounding the markers
        QPen painterPen( pen );
        painterPen.setStyle( Qt::SolidLine );
        painter->setPen( painterPen );
        painter->setBrush( brush );
        painter->setRenderHint ( QPainter::Antialiasing );
        painter->translate( pos );
        switch ( markerAttributes.markerStyle() ) {
            case MarkerAttributes::MarkerCircle:
                painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                            maSize.height(), maSize.width()) );
                break;
            case MarkerAttributes::MarkerSquare:
                {
                    QRectF rect( 0 - maSize.width()/2, 0 - maSize.height()/2,
                                maSize.width(), maSize.height() );
                    painter->drawRect( rect );
                    painter->fillRect( rect, brush.color() );
                    break;
                }
            case MarkerAttributes::MarkerDiamond:
                {
                    QVector <QPointF > diamondPoints;
                    QPointF top, left, bottom, right;
                    top    = QPointF( 0, 0 - maSize.height()/2 );
                    left   = QPointF( 0 - maSize.width()/2, 0 );
                    bottom = QPointF( 0, maSize.height()/2 );
                    right  = QPointF( maSize.width()/2, 0 );
                    diamondPoints << top << left << bottom << right;
                    painter->drawPolygon( diamondPoints );
                    break;
                }
            // both handled on top of the method:
            case MarkerAttributes::Marker1Pixel:
            case MarkerAttributes::Marker4Pixels:
                    break;
            case MarkerAttributes::MarkerRing:
                {
                    painter->setPen( QPen( brush.color() ) );
                    painter->setBrush( Qt::NoBrush );
                    painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                                        maSize.height(), maSize.width()) );
                    break;
                }
            case MarkerAttributes::MarkerCross:
                {
                    QRectF rect( maSize.width()*-0.5, maSize.height()*-0.2,
                                 maSize.width(), maSize.height()*0.4 );
                    painter->drawRect( rect );
                    rect.setTopLeft(QPointF( maSize.width()*-0.2, maSize.height()*-0.5 ));
                    rect.setSize(QSizeF( maSize.width()*0.4, maSize.height() ));
                    painter->drawRect( rect );
                    break;
                }
            case MarkerAttributes::MarkerFastCross:
                {
                    QPointF left, right, top, bottom;
                    left  = QPointF( -maSize.width()/2, 0 );
                    right = QPointF( maSize.width()/2, 0 );
                    top   = QPointF( 0, -maSize.height()/2 );
                    bottom= QPointF( 0, maSize.height()/2 );
                    painter->setPen( QPen( brush.color() ) );
                    painter->drawLine( left, right );
                    painter->drawLine(  top, bottom );
                    break;
                }
            default:
                Q_ASSERT_X ( false, "paintMarkers()",
                            "Type item does not match a defined Marker Type." );
        }
    }
    painter->setPen( oldPen );
}

void AbstractDiagram::paintMarkers( QPainter* painter )
```

7.17.3.49 void AbstractDiagram::paintMarkers (QPainter * *painter*) [protected, virtual, inherited]

Definition at line 716 of file KDChartAbstractDiagram.cpp.

```
                                                                               {
       for ( int j=0; j< rowCount; ++j ) {
           const QModelIndex index = model()->index( j, i, rootIndex() );
           double value = model()->data( index ).toDouble();
           const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
           paintMarker( painter, index, pos );
       }
    }
}


```

7.17.3.50 QPen AbstractDiagram::pen (const QModelIndex & *index*) const [inherited]

Retrieve the pen to be used, for painting the datapoint at the given index in the model.

Parameters:

index The index of the datapoint in the model.

Returns:

The pen to use for painting.

Definition at line 770 of file KDChartAbstractDiagram.cpp.

```
777 {
```

7.17.3.51 QPen AbstractDiagram::pen (int *dataset*) const `[inherited]`

Retrieve the pen to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the pen for.

Returns:

The pen to use for painting.

Definition at line 762 of file KDChartAbstractDiagram.cpp.

```
769 {
```

7.17.3.52 QPen AbstractDiagram::pen () const `[inherited]`

Retrieve the pen to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The pen to use for painting.

Definition at line 756 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::PieDiagram::paint(), and KDChart::LineDiagram::paint().

```
761 {
```

7.17.3.53 bool AbstractDiagram::percentMode () const `[inherited]`

Definition at line 468 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::CartesianCoordinatePlane::getDataDimensionsList().

7.17.3.54 void KDChart::AbstractDiagram::propertiesChanged () `[protected, inherited]`

Emitted upon change of a property of the Diagram.

Referenced by KDChart::LineDiagram::resetLineAttributes(), KDChart::AbstractDiagram::setDataValueAttributes(), KDChart::LineDiagram::setLineAttributes(), KDChart::LineDiagram::setThreeDLineAttributes(), and KDChart::LineDiagram::setType().

7.17.3.55 AbstractCartesianDiagram ∗ AbstractCartesianDiagram::referenceDiagram () const `[virtual, inherited]`

Definition at line 146 of file KDChartAbstractCartesianDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::paint(), and referenceDiagramIsBarDiagram().

```
{
    return d->referenceDiagram;
}
```

7.17.3.56 QPointF AbstractCartesianDiagram::referenceDiagramOffset () const `[virtual, inherited]`

Definition at line 151 of file KDChartAbstractCartesianDiagram.cpp.

References d.

```
{
    return d->referenceDiagramOffset;
}
```

7.17.3.57 void BarDiagram::resize (const QSizeF & *area*) `[virtual]`

Called by the widget's sizeEvent.

Adjust all internal structures, that are calculated, dependending on the size of the widget.

Parameters:

area

Implements KDChart::AbstractDiagram.

Definition at line 670 of file KDChartBarDiagram.cpp.

```
{
}
```

7.17.3.58 void BarDiagram::resizeEvent (QResizeEvent ∗) `[protected]`

Definition at line 193 of file KDChartBarDiagram.cpp.

```
{

}
```

7.17.3.59 void AbstractDiagram::scrollTo (const QModelIndex & *index*, ScrollHint *hint* = EnsureVisible) `[virtual, inherited]`

[reimplemented]

Definition at line 830 of file KDChartAbstractDiagram.cpp.

```
832 { return QModelIndex(); }
```

7.17.3.60 void AbstractDiagram::setAllowOverlappingDataValueTexts (bool *allow*) `[inherited]`

Set whether data value labels are allowed to overlap.

Parameters:

allow True means that overlapping labels are allowed.

Definition at line 440 of file KDChartAbstractDiagram.cpp.

References d.

```
445 {
```

7.17.3.61 void AbstractDiagram::setAntiAliasing (bool *enabled*) `[inherited]`

Set whether anti-aliasing is to be used while rendering this diagram.

Parameters:

enabled True means that AA is enabled.

Definition at line 451 of file KDChartAbstractDiagram.cpp.

References d.

```
456 {
```

7.17.3.62 void AbstractDiagram::setAttributesModel (AttributesModel ∗ *model*) `[virtual, inherited]`

Associate an AttributesModel with this diagram.

Note that the diagram does _not_ take ownership of the AttributesModel. This should thus only be used with AttributesModels that have been explicitely created by the user, and are owned by her. Setting an AttributesModel that is internal to another diagram is an error.

Correct:

```
AttributesModel *am = new AttributesModel( model, 0 );
diagram1->setAttributesModel( am );
diagram2->setAttributesModel( am );
```

Wrong:

```
diagram1->setAttributesModel( diagram2->attributesModel() );
```

Parameters:

model The AttributesModel to use for this diagram.

See also:

AttributesModel, usesExternalAttributesModel

Definition at line 261 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::AbstractDiagram::modelsChanged().

```
{
    if( amodel->sourceModel() != model() ) {
        qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
                 "Trying to set an attributesmodel which works on a different "
                 "model than the diagram.");
        return;
    }
    if( qobject_cast<PrivateAttributesModel*>(amodel) ) {
        qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
                 "Trying to set an attributesmodel that is private to another diagram.");
        return;
    }
    d->setAttributesModel(amodel);
    scheduleDelayedItemsLayout();
    d->databoundariesDirty = true;
    emit modelsChanged();
}
```

7.17.3.63 void AbstractDiagram::setAttributesModelRootIndex (const QModelIndex & *idx*) `[protected, inherited]`

Definition at line 301 of file KDChartAbstractDiagram.cpp.

References d.

7.17.3.64 void BarDiagram::setBarAttributes (const QModelIndex & *index*, const BarAttributes & *a*)

Definition at line 100 of file KDChartBarDiagram.cpp.

References KDChart::AbstractDiagram::attributesModel(), KDChart::BarAttributesRole, d, and KDChart::AttributesModel::setData().

```
{
    attributesModel()->setData(
        d->attributesModel->mapFromSource( index ),
        qVariantFromValue( ta ),
        BarAttributesRole );
}
```

7.17.3.65 void BarDiagram::setBarAttributes (int *column*, const BarAttributes & *a*)

Definition at line 92 of file KDChartBarDiagram.cpp.

References KDChart::BarAttributesRole, and d.

```
{
    d->attributesModel->setHeaderData(
        column, Qt::Vertical,
        qVariantFromValue( ta ),
        BarAttributesRole );
}
```

7.17.3.66 void BarDiagram::setBarAttributes (const BarAttributes & *a*)

Definition at line 87 of file KDChartBarDiagram.cpp.

References KDChart::BarAttributesRole, and d.

```
{
    d->attributesModel->setModelData( qVariantFromValue( ta ), BarAttributesRole );
}
```

7.17.3.67 void AbstractDiagram::setBrush (const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting all datasets in the model.

Parameters:

brush The brush to use.

Definition at line 786 of file KDChartAbstractDiagram.cpp.

```
{
```

7.17.3.68 void AbstractDiagram::setBrush (int *dataset*, const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting the given dataset.

Parameters:

dataset The dataset's column in the model.

pen The brush to use.

Definition at line 793 of file KDChartAbstractDiagram.cpp.

```
{
```

7.17.3.69 void AbstractDiagram::setBrush (const QModelIndex & *index*, const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting the datapoint at the given index.

Parameters:

index The datapoint's index in the model.

brush The brush to use.

Definition at line 778 of file KDChartAbstractDiagram.cpp.

```
{
```

7.17.3.70 void KDChart::AbstractCartesianDiagram::setCoordinatePlane (AbstractCoordinatePlane ∗ *plane*) `[virtual, inherited]`

Set the coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. The chart takes ownership.

Returns:

The coordinate plane associated with the diagram.

Reimplemented from KDChart::AbstractDiagram.

Definition at line 123 of file KDChartAbstractCartesianDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane(), and KDChart::AbstractDiagram::setCoordinatePlane().

```
{
    if( coordinatePlane() ) disconnect( coordinatePlane() );
    AbstractDiagram::setCoordinatePlane(plane);

    // show the axes, after all have been adjusted
    // (because they might be dependend on each other)
    /*
    if( plane )
        Q_FOREACH( CartesianAxis* axis, d->axesList )
            axis->show();
    else
        Q_FOREACH( CartesianAxis* axis, d->axesList )
            axis->hide();
    */
}
```

7.17.3.71 void AbstractDiagram::setDataBoundariesDirty () const `[protected, inherited]`

Definition at line 240 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by setThreeDBarAttributes(), KDChart::LineDiagram::setThreeDLineAttributes(), KDChart::LineDiagram::setType(), and setType().

```
{
    d->databoundariesDirty = true;
}
```

7.17.3.72 void AbstractDiagram::setDatasetDimension (int *dimension*) `[inherited]`

Sets the dataset dimension of the diagram.

See also:

datasetDimension.

Parameters:

dimension

Definition at line 947 of file KDChartAbstractDiagram.cpp.

References d.

```
954 {
```

7.17.3.73 void AbstractDiagram::setDataValueAttributes (const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for all datapoints in the model.

Parameters:

a The attributes to set.

Definition at line 434 of file KDChartAbstractDiagram.cpp.

References d.

```
439 {
```

7.17.3.74 void AbstractDiagram::setDataValueAttributes (int *dataset*, const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for the given dataset.

Parameters:

dataset The dataset to set the attributes for.

a The attributes to set.

Definition at line 406 of file KDChartAbstractDiagram.cpp.

References d.

```
413 {
```

7.17.3.75 void AbstractDiagram::setDataValueAttributes (const QModelIndex & *index*, const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for the given index.

Parameters:

index The datapoint to set the attributes for.

a The attributes to set.

Definition at line 395 of file KDChartAbstractDiagram.cpp.

References d, KDChart::DataValueLabelAttributesRole, and KDChart::AbstractDiagram::propertiesChanged().

```
{
    d->attributesModel->setData(
        d->attributesModel->mapFromSource( index ),
        qVariantFromValue( a ),
        DataValueLabelAttributesRole );
    emit propertiesChanged();
}


```

7.17.3.76 void AbstractDiagram::setHidden (bool *hidden*) `[inherited]`

Hide (or unhide, resp.) all datapoints in the model.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

hidden The hidden status to set.

Definition at line 365 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.17.3.77 void AbstractDiagram::setHidden (int *column*, bool *hidden*) `[inherited]`

Hide (or unhide, resp.) a dataset.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

dataset The dataset to set the hidden status for.

hidden The hidden status to set.

Definition at line 356 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.17.3.78 void AbstractDiagram::setHidden (const QModelIndex & *index*, bool *hidden*) `[inherited]`

Hide (or unhide, resp.) a data cell.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

index The datapoint to set the hidden status for.

hidden The hidden status to set.

Definition at line 347 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::DataHiddenRole.

```
{
```

7.17.3.79 void AbstractDiagram::setModel (QAbstractItemModel ∗ *model*) `[virtual, inherited]`

Associate a model with the diagram.

Definition at line 245 of file KDChartAbstractDiagram.cpp.

References d, KDChart::AttributesModel::initFrom(), and KDChart::AbstractDiagram::modelsChanged().

```
{
  QAbstractItemView::setModel( newModel );
  AttributesModel* amodel = new PrivateAttributesModel( newModel, this );
  amodel->initFrom( d->attributesModel );
  d->setAttributesModel(amodel);
  scheduleDelayedItemsLayout();
  d->databoundariesDirty = true;
  emit modelsChanged();
}
```

7.17.3.80 void AbstractDiagram::setPen (const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting all datasets in the model.

Parameters:

pen The pen to use.

Definition at line 740 of file KDChartAbstractDiagram.cpp.

```
{
```

7.17.3.81 void AbstractDiagram::setPen (int *dataset*, const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting the given dataset.

Parameters:

- ***dataset*** The dataset's row in the model.
- ***pen*** The pen to use.

Definition at line 747 of file KDChartAbstractDiagram.cpp.

```
755 {
```

7.17.3.82 void AbstractDiagram::setPen (const QModelIndex & *index*, const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting the datapoint at the given index.

Parameters:

- ***index*** The datapoint's index in the model.
- ***pen*** The pen to use.

Definition at line 732 of file KDChartAbstractDiagram.cpp.

```
739 {
```

7.17.3.83 void AbstractDiagram::setPercentMode (bool *percent*) `[inherited]`

Definition at line 462 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::setType(), and setType().

```
467 {
```

7.17.3.84 void AbstractCartesianDiagram::setReferenceDiagram (AbstractCartesianDiagram ∗ *diagram*, const QPointF & *offset* = QPointF()) `[virtual, inherited]`

Definition at line 140 of file KDChartAbstractCartesianDiagram.cpp.

References d.

```
141 {
142     d->referenceDiagram = diagram;
143     d->referenceDiagramOffset = offset;
144 }
```

7.17.3.85 void AbstractDiagram::setRootIndex (const QModelIndex & *idx*) `[virtual, inherited]`

Set the root index in the model, where the diagram starts referencing data for display.

[reimplemented]

Definition at line 294 of file KDChartAbstractDiagram.cpp.

References d.

7.17.3.86 void AbstractDiagram::setSelection (const QRect & *rect*, QItemSelectionModel::SelectionFlags *command*) `[virtual, inherited]`

[reimplemented]

Definition at line 848 of file KDChartAbstractDiagram.cpp.

```
{ return QRegion(); }
```

7.17.3.87 void BarDiagram::setThreeDBarAttributes (const QModelIndex & *index*, const ThreeDBarAttributes & *a*)

Definition at line 147 of file KDChartBarDiagram.cpp.

References d, KDChart::AbstractDiagram::setDataBoundariesDirty(), and KDChart::ThreeDBarAttributesRole.

```
{
    setDataBoundariesDirty();
    d->attributesModel->setData(
        d->attributesModel->mapFromSource(index),
        qVariantFromValue( threeDAttrs ),
        ThreeDBarAttributesRole );
    emit layoutChanged( this );
}
```

7.17.3.88 void BarDiagram::setThreeDBarAttributes (int *column*, const ThreeDBarAttributes & *a*)

Definition at line 137 of file KDChartBarDiagram.cpp.

References d, KDChart::AbstractDiagram::setDataBoundariesDirty(), and KDChart::ThreeDBarAttributesRole.

```
{
    setDataBoundariesDirty();
    d->attributesModel->setHeaderData(
        column, Qt::Vertical,
        qVariantFromValue( threeDAttrs ),
        ThreeDBarAttributesRole );
    emit layoutChanged( this );
}
```

7.17.3.89 void BarDiagram::setThreeDBarAttributes (const ThreeDBarAttributes & *a*)

Definition at line 130 of file KDChartBarDiagram.cpp.

References d, KDChart::AbstractDiagram::setDataBoundariesDirty(), and KDChart::ThreeDBarAttributesRole.

```
{
    setDataBoundariesDirty();
    d->attributesModel->setModelData( qVariantFromValue( threeDAttrs ), ThreeDBarAttributesRole );
    emit layoutChanged( this );
}
```

7.17.3.90 void BarDiagram::setType (BarType *type*)

Definition at line 71 of file KDChartBarDiagram.cpp.

References d, KDChart::AbstractDiagram::setDataBoundariesDirty(), and KDChart::AbstractDiagram::setPercentMode().

```
{
   if ( type == d->barType ) return;

   d->barType = type;
   // AbstractAxis settings - see AbstractDiagram and CartesianAxis
   setPercentMode( type == BarDiagram::Percent );
   setDataBoundariesDirty();
   emit layoutChanged( this );
}
```

7.17.3.91 void AbstractCartesianDiagram::takeAxis (CartesianAxis ∗ *axis*) `[virtual, inherited]`

Removes the axis from the diagram, without deleting it.

The diagram no longer owns the axis, so it is the caller's responsibility to delete the axis.

See also:
addAxis

Definition at line 98 of file KDChartAbstractCartesianDiagram.cpp.

References d, KDChart::AbstractAxis::deleteObserver(), KDChart::AbstractCartesianDiagram::layoutPlanes(), and KDChart::AbstractLayoutItem::setParentWidget().

Referenced by KDChart::CartesianAxis::~CartesianAxis().

```
{
    const int idx = d->axesList.indexOf( axis );
    if( idx != -1 )
        d->axesList.takeAt( idx );
    axis->deleteObserver( this );
    axis->setParentWidget( 0 );
    layoutPlanes();
}
```

7.17.3.92 ThreeDBarAttributes BarDiagram::threeDBarAttributes (const QModelIndex & *index*) const

Definition at line 171 of file KDChartBarDiagram.cpp.

References d.

```
{
    return qVariantValue<ThreeDBarAttributes>(
        d->attributesModel->data(
            d->attributesModel->mapFromSource(index),
            KDChart::ThreeDBarAttributesRole ) );
}
```

7.17.3.93 ThreeDBarAttributes BarDiagram::threeDBarAttributes (int *column*) const

Definition at line 163 of file KDChartBarDiagram.cpp.

References d.

```
{
    return qVariantValue<ThreeDBarAttributes>(
        d->attributesModel->data(
            d->attributesModel->mapFromSource( columnToIndex( column ) ),
            KDChart::ThreeDBarAttributesRole ) );
}
```

7.17.3.94 ThreeDBarAttributes BarDiagram::threeDBarAttributes () const

Definition at line 157 of file KDChartBarDiagram.cpp.

References d.

Referenced by paint(), and threeDItemDepth().

```
{
    return qVariantValue<ThreeDBarAttributes>(
        d->attributesModel->data( KDChart::ThreeDBarAttributesRole ) );
}
```

7.17.3.95 double BarDiagram::threeDItemDepth (int *column*) const `[protected, virtual]`

Implements KDChart::AbstractCartesianDiagram.

Definition at line 184 of file KDChartBarDiagram.cpp.

References d.

```
{
    return qVariantValue<ThreeDBarAttributes>(
        d->attributesModel->headerData (
            column,
            Qt::Vertical,
            KDChart::ThreeDBarAttributesRole ) ).validDepth();
}
```

7.17.3.96 double BarDiagram::threeDItemDepth (const QModelIndex & *index*) const `[protected, virtual]`

Implements KDChart::AbstractCartesianDiagram.

Definition at line 179 of file KDChartBarDiagram.cpp.

References threeDBarAttributes(), and KDChart::AbstractThreeDAttributes::validDepth().

```
180 {
181     return threeDBarAttributes( index ).validDepth();
182 }
```

7.17.3.97 BarDiagram::BarType BarDiagram::type () const

Definition at line 82 of file KDChartBarDiagram.cpp.

References d.

Referenced by calculateDataBoundaries(), and paint().

```
83 {
84    return d->barType;
85 }
```

7.17.3.98 void AbstractDiagram::update () const `[inherited]`

Definition at line 961 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractDiagram::doItemsLayout().

7.17.3.99 void KDChart::AbstractDiagram::useDefaultColors () `[inherited]`

Set the palette to be used, for painting datasets to the default palette.

See also:

KDChart::Palette. FIXME: fold into one usePalette (KDChart::Palette&) method

Definition at line 855 of file KDChartAbstractDiagram.cpp.

References d.

```
859 {
```

7.17.3.100 void KDChart::AbstractDiagram::useRainbowColors () `[inherited]`

Set the palette to be used, for painting datasets to the rainbow palette.

See also:

KDChart::Palette.

Definition at line 865 of file KDChartAbstractDiagram.cpp.

References d.

```
869 {
```

7.17.3.101 bool AbstractDiagram::usesExternalAttributesModel () const `[virtual, inherited]`

Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

See also:
setAttributesModel

Definition at line 280 of file KDChartAbstractDiagram.cpp.

References d.

```
281 {
282     return d->usesExternalAttributesModel();
283 }
```

7.17.3.102 void KDChart::AbstractDiagram::useSubduedColors () `[inherited]`

Set the palette to be used, for painting datasets to the subdued palette.

See also:
KDChart::Palette.

Definition at line 860 of file KDChartAbstractDiagram.cpp.

References d.

```
864 {
```

7.17.3.103 double AbstractDiagram::valueForCell (int *row*, int *column*) const `[protected, inherited]`

Helper method, retrieving the data value (DisplayRole) for a given row and column.

Parameters:

row The row to query.

column The column to query.

Returns:
The value of the display role at the given row and column as a double.

Definition at line 955 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::attributesModelRootIndex(), and d.

Referenced by KDChart::LineDiagram::paint().

```
960 {
```

7.17.3.104 int AbstractDiagram::verticalOffset () const `[virtual, inherited]`

[reimplemented]

Definition at line 842 of file KDChartAbstractDiagram.cpp.

```
844 { return true; }
```

7.17.3.105 QRect AbstractDiagram::visualRect (const QModelIndex & *index*) const `[virtual, inherited]`

[reimplemented]

Definition at line 825 of file KDChartAbstractDiagram.cpp.

```
829 {}
```

7.17.3.106 QRegion AbstractDiagram::visualRegionForSelection (const QItemSelection & *selection*) const `[virtual, inherited]`

[reimplemented]

Definition at line 851 of file KDChartAbstractDiagram.cpp.

7.17.4 Member Data Documentation

7.17.4.1 Q_SIGNALS KDChart::AbstractDiagram::__pad0__ `[protected, inherited]`

Definition at line 589 of file KDChartAbstractDiagram.h.

The documentation for this class was generated from the following files:

- KDChartBarDiagram.h
- KDChartBarDiagram.cpp

7.18 KDChart::CartesianAxis Class Reference

`#include <KDChartCartesianAxis.h>`

Inheritance diagram for KDChart::CartesianAxis:Collaboration diagram for KDChart::CartesianAxis:

7.18.1 Detailed Description

The class for cartesian axes.

For being useful, axes need to be assigned to a diagram, see AbstractCartesianDiagram::addAxis and AbstractCartesianDiagram::takeAxis.

See also:
PolarAxis, AbstractCartesianDiagram

Definition at line 48 of file KDChartCartesianAxis.h.

Public Types

- enum Position {
  Bottom,
  Top,
  Right,
  Left }

Public Member Functions

- void alignToReferencePoint (const RelativePosition &position)
- BackgroundAttributes backgroundAttributes () const
- virtual int bottomOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart:AutoSpacerLayoutItem::sizeHint().

- CartesianAxis (AbstractCartesianDiagram ∗diagram=0)

  C'tor of the class for cartesian axes.

- bool compare (const AbstractAreaBase ∗other) const

  Returns true if both areas have the same settings.

- bool compare (const AbstractAxis ∗other) const

  Returns true if both axes have the same settings.

- bool compare (const CartesianAxis ∗other) const

  Returns true if both axes have the same settings.

- virtual void connectSignals ()

  Wireing the signal/slot connections.

- const AbstractCoordinatePlane ∗ coordinatePlane () const

Convenience function, returns the coordinate plane, in which this axis is used.

- void createObserver (AbstractDiagram ∗diagram)
- virtual const QString customizedLabel (const QString &label) const

  Implement this method if you want to adjust axis labels before they are printed.

- void deleteObserver (AbstractDiagram ∗diagram)
- const AbstractDiagram ∗ diagram () const
- virtual Qt::Orientations expandingDirections () const

  pure virtual in QLayoutItem

- FrameAttributes frameAttributes () const
- virtual QRect geometry () const

  pure virtual in QLayoutItem

- void getFrameLeadings (int &left, int &top, int &right, int &bottom) const
- bool hasDefaultTitleTextAttributes () const
- virtual bool isAbscissa () const
- virtual bool isEmpty () const

  pure virtual in QLayoutItem

- virtual bool isOrdinate () const
- QStringList labels () const

  Returns a list of strings, that are used as axis labels, as set via setLabels.

- virtual void layoutPlanes ()
- virtual int leftOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- virtual QSize maximumSize () const

  pure virtual in QLayoutItem

- virtual QSize minimumSize () const

  pure virtual in QLayoutItem

- bool observedBy (AbstractDiagram ∗diagram) const
- virtual void paint (QPainter ∗)

  reimpl

- virtual void paintAll (QPainter &painter)

  Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

- virtual void paintBackground (QPainter &painter, const QRect &rectangle)
- virtual void paintCtx (PaintContext ∗)

  reimpl

- virtual void paintFrame (QPainter &painter, const QRect &rectangle)
- virtual void paintIntoRect (QPainter &painter, const QRect &rect)

  Draws the background and frame, then calls paint().

- QLayout ∗ parentLayout ()
- virtual const Position position () const
- void removeFromParentLayout ()
- void resetTitleTextAttributes ()

  Reset the title text attributes to the built-in default:.

- virtual int rightOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- void setBackgroundAttributes (const BackgroundAttributes &a)
- void setFrameAttributes (const FrameAttributes &a)
- virtual void setGeometry (const QRect &r)

  pure virtual in QLayoutItem

- void setLabels (const QStringList &list)

  Use this to specify your own set of strings, to be used as axis labels.

- void setParentLayout (QLayout ∗lay)
- virtual void setParentWidget (QWidget ∗widget)

  Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

- virtual void setPosition (Position p)
- void setShortLabels (const QStringList &list)

  Use this to specify your own set of strings, to be used as axis labels, in case the normal labels are too long.

- void setTextAttributes (const TextAttributes &a)

  Use this to specify the text attributes to be used for axis labels.

- void setTitleText (const QString &text)
- void setTitleTextAttributes (const TextAttributes &a)
- QStringList shortLabels () const

  Returns a list of strings, that are used as axis labels, as set via setShortLabels.

- virtual QSize sizeHint () const

  pure virtual in QLayoutItem

- virtual void sizeHintChanged () const

  Report changed size hint: ask the parent widget to recalculate the layout.

- TextAttributes textAttributes () const

  Returns the text attributes to be used for axis labels.

- int tickLength (bool subUnitTicks=false) const
- QString titleText () const
- TextAttributes titleTextAttributes () const

  Returns the text attributes that will be used for displaying the title text.

- virtual int topOverlap (bool doNotRecalculate=false) const

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- ∼CartesianAxis ()

Static Public Member Functions

- void paintBackgroundAttributes (QPainter &painter, const QRect &rectangle, const KDChart::BackgroundAttributes &attributes)
- void paintFrameAttributes (QPainter &painter, const QRect &rectangle, const KDChart::FrameAttributes &attributes)

Public Attributes

- public Q_SLOTS: void update()
- protected Q_SLOTS: virtual void delayedInit()

Protected Member Functions

- virtual QRect areaGeometry () const
- QRect innerRect () const
- virtual void positionHasChanged ()

Protected Attributes

- Q_SIGNALS __pad0__: void positionChanged(AbstractArea ∗)
- QWidget ∗ mParent
- QLayout ∗ mParentLayout

7.18.2 Member Enumeration Documentation

7.18.2.1 enum KDChart::CartesianAxis::Position

Enumeration values:

Bottom

Top

Right

Left

Definition at line 56 of file KDChartCartesianAxis.h.

```
56                        {
57              Bottom,
58              Top,
59              Right,
60              Left
61          };
```

7.18.3 Constructor & Destructor Documentation

7.18.3.1 CartesianAxis::CartesianAxis (AbstractCartesianDiagram ∗ *diagram* = 0) `[explicit]`

C'tor of the class for cartesian axes.

Note:

If using a zero parent for the constructor, you need to call your diagram's addAxis function to add your axis to the diagram. Otherwise, there is no need to call addAxis, since the constructor does that automatically for you, if you pass a diagram as parameter.

See also:

AbstractCartesianDiagram::addAxis

Definition at line 51 of file KDChartCartesianAxis.cpp.

```
52     : AbstractAxis ( new Private( diagram, this ), diagram )
53 {
54     init();
55 }
```

7.18.3.2 CartesianAxis::∼CartesianAxis ()

Definition at line 57 of file KDChartCartesianAxis.cpp.

References d, and KDChart::AbstractCartesianDiagram::takeAxis().

```
58 {
59     // when we remove the first axis it will unregister itself and
60     // propagate the next one to the primary, thus the while loop
61     while ( d->mDiagram ) {
62         AbstractCartesianDiagram *cd = qobject_cast<AbstractCartesianDiagram*>( d->mDiagram );
63         cd->takeAxis( this );
64     }
65     Q_FOREACH( AbstractDiagram *diagram, d->secondaryDiagrams ) {
66         AbstractCartesianDiagram *cd = qobject_cast<AbstractCartesianDiagram*>( diagram );
67         cd->takeAxis( this );
68     }
69 }
```

7.18.4 Member Function Documentation

7.18.4.1 void AbstractAreaBase::alignToReferencePoint (const RelativePosition & *position*) `[inherited]`

Definition at line 90 of file KDChartAbstractAreaBase.cpp.

```
91 {
92     Q_UNUSED( position );
93     // PENDING(kalle) FIXME
94     qWarning( "Sorry, not implemented: void AbstractAreaBase::alignToReferencePoint( const RelativePosi
95 }
```

7.18.4.2 QRect AbstractArea::areaGeometry () const `[protected, virtual, inherited]`

Implements KDChart::AbstractAreaBase.

Definition at line 150 of file KDChartAbstractArea.cpp.

Referenced by KDChart::CartesianCoordinatePlane::drawingArea(), KDChart::PolarCoordinatePlane::layoutDiagrams(), paint(), KDChart::AbstractArea::paintAll(), and paintCtx().

```
{
    return geometry();
}
```

7.18.4.3 BackgroundAttributes AbstractAreaBase::backgroundAttributes () const `[inherited]`

Definition at line 112 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by updateCommonBrush().

```
{
    return d->backgroundAttributes;
}
```

7.18.4.4 int AbstractArea::bottomOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart:AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the bottom edge of the area.

Note:

> The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 101 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
    // Re-calculate the sizes,
    // so we also get the amountOf..Overlap members set newly:
    if( ! doNotRecalculate )
        sizeHint();
    return d->amountOfBottomOverlap;
}
```

7.18.4.5 bool AbstractAreaBase::compare (const AbstractAreaBase ∗ *other*) const `[inherited]`

Returns true if both areas have the same settings.

Definition at line 75 of file KDChartAbstractAreaBase.cpp.

```
{
    if( other == this ) return true;
    if( ! other ){
        //qDebug() << "CartesianAxis::compare() cannot compare to Null pointer";
        return false;
    }
    /*
    qDebug() << "AbstractAreaBase:" << (frameAttributes() == other->frameAttributes())
        << (backgroundAttributes() == other->backgroundAttributes()) << "\n";
    */
    return  (frameAttributes()      == other->frameAttributes()) &&
            (backgroundAttributes() == other->backgroundAttributes());
}
```

7.18.4.6 bool AbstractAxis::compare (const AbstractAxis ∗ *other*) const `[inherited]`

Returns true if both axes have the same settings.

Definition at line 142 of file KDChartAbstractAxis.cpp.

```
{
    if( other == this ) return true;
    if( ! other ){
        //qDebug() << "CartesianAxis::compare() cannot compare to Null pointer";
        return false;
    }
    /*
    qDebug() << (textAttributes() == other->textAttributes());
    qDebug() << (labels()         == other->labels());
    qDebug() << (shortLabels()    == other->shortLabels());
    */
    return  ( static_cast<const AbstractAreaBase*>(this)->compare( other ) ) &&
            (textAttributes() == other->textAttributes()) &&
            (labels()         == other->labels()) &&
            (shortLabels()    == other->shortLabels());
}
```

7.18.4.7 bool CartesianAxis::compare (const CartesianAxis ∗ *other*) const

Returns true if both axes have the same settings.

Definition at line 77 of file KDChartCartesianAxis.cpp.

```
{
    if( other == this ) return true;
    if( ! other ){
        //qDebug() << "CartesianAxis::compare() cannot compare to Null pointer";
        return false;
    }
    /*
    qDebug() << (position()            == other->position());
    qDebug() << (titleText()           == other->titleText());
    qDebug() << (titleTextAttributes() == other->titleTextAttributes());
88      */
89      return  ( static_cast<const AbstractAxis*>(this)->compare( other ) ) &&
90              ( position()            == other->position() ) &&
91              ( titleText()           == other->titleText() ) &&
92              ( titleTextAttributes() == other->titleTextAttributes() );
93 }
```

7.18.4.8 void AbstractAxis::connectSignals () `[virtual, inherited]`

Wireing the signal/slot connections.

This method gets called automatically, each time, when you assign the axis to a diagram, either by passing a diagram∗ to the c'tor, or by calling the diagram's setAxis method, resp.

If overwriting this method in derived classes, make sure to call this base method AbstractAxis::connectSignals(), so your axis gets connected to the diagram's built-in signals.

See also:
AbstractCartesianDiagram::addAxis()

Definition at line 211 of file KDChartAbstractAxis.cpp.

References d.

Referenced by KDChart::AbstractAxis::createObserver().

```
212 {
213     if( d->observer ){
214         connect( d->observer, SIGNAL( diagramDataChanged( AbstractDiagram *) ),
215                 this, SLOT( update() ) );
216     }
217 }
```

7.18.4.9 const AbstractCoordinatePlane ∗ AbstractAxis::coordinatePlane () const `[inherited]`

Convenience function, returns the coordinate plane, in which this axis is used.

If the axis is not used in a coordinate plane, the return value is Zero.

Definition at line 312 of file KDChartAbstractAxis.cpp.

References d.

```
313 {
314     if( d->diagram() )
315         return d->diagram()->coordinatePlane();
316     return 0;
317 }
```

7.18.4.10 void AbstractAxis::createObserver (AbstractDiagram ∗ *diagram*) `[inherited]`

Definition at line 177 of file KDChartAbstractAxis.cpp.

References KDChart::AbstractAxis::connectSignals(), and d.

Referenced by KDChart::AbstractCartesianDiagram::addAxis().

```
178 {
179     if( d->setDiagram( diagram ) )
180         connectSignals();
181 }
```

7.18.4.11 const QString AbstractAxis::customizedLabel (const QString & *label*) const `[virtual, inherited]`

Implement this method if you want to adjust axis labels before they are printed.

KD Chart is calling this method immediately before drawing the text, this means: What you return here will be drawn without further modifications.

Parameters:

label The text of the label as KD Chart has calculated it automatically (or as it was taken from a QStringList provided by you, resp.)

Returns:

The text to be drawn. By default this is the same as `label`.

Definition at line 161 of file KDChartAbstractAxis.cpp.

Referenced by maximumSize(), and paintCtx().

```
162 {
163     return label;
164 }
```

7.18.4.12 void AbstractAxis::deleteObserver (AbstractDiagram ∗ *diagram*) `[inherited]`

Definition at line 193 of file KDChartAbstractAxis.cpp.

References d.

Referenced by KDChart::AbstractCartesianDiagram::takeAxis(), and KDChart::AbstractCartesianDiagram::∼AbstractCartesianDiagram().

```
194 {
195     d->unsetDiagram( diagram );
196 }
```

7.18.4.13 const AbstractDiagram ∗ KDChart::AbstractAxis::diagram () const `[inherited]`

Definition at line 319 of file KDChartAbstractAxis.cpp.

References d.

```
320 {
321     return d->diagram();
322 }
```

7.18.4.14 Qt::Orientations CartesianAxis::expandingDirections () const `[virtual]`

pure virtual in QLayoutItem

Definition at line 960 of file KDChartCartesianAxis.cpp.

References Bottom, Left, position(), Right, and Top.

```
{
    Qt::Orientations ret;
    switch ( position() )
    {
    case Bottom:
    case Top:
        ret = Qt::Horizontal;
        break;
    case Left:
    case Right:
        ret = Qt::Vertical;
        break;
    default:
        Q_ASSERT( false ); // all positions need to be handeld
        break;
    };
    return ret;
}
```

7.18.4.15 FrameAttributes AbstractAreaBase::frameAttributes () const `[inherited]`

Definition at line 102 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone(), and updateCommonBrush().

```
{
    return d->frameAttributes;
}
```

7.18.4.16 QRect CartesianAxis::geometry () const `[virtual]`

pure virtual in QLayoutItem

Implements KDChart::AbstractAxis.

Definition at line 1197 of file KDChartCartesianAxis.cpp.

References d.

Referenced by paintCtx().

```
{
    return d->geometry;
}
```

7.18.4.17 void AbstractAreaBase::getFrameLeadings (int & *left*, int & *top*, int & *right*, int & *bottom*) const `[inherited]`

Definition at line 204 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::AbstractAreaBase::innerRect(), and KDChart::AbstractAreaWidget::paintAll().

```
{
    if( d && d->frameAttributes.isVisible() ){
        const int padding = qMax( d->frameAttributes.padding(), 0 );
        left   = padding;
        top    = padding;
        right  = padding;
        bottom = padding;
    }else{
        left   = 0;
        top    = 0;
        right  = 0;
        bottom = 0;
    }
}
```

7.18.4.18 bool CartesianAxis::hasDefaultTitleTextAttributes () const

Definition at line 133 of file KDChartCartesianAxis.cpp.

References d.

Referenced by titleTextAttributes().

```
{
    return d->useDefaultTextAttributes;
}
```

7.18.4.19 QRect AbstractAreaBase::innerRect () const `[protected, inherited]`

Definition at line 220 of file KDChartAbstractAreaBase.cpp.

References KDChart::AbstractAreaBase::areaGeometry(), and KDChart::AbstractAreaBase::getFrameLeadings().

Referenced by KDChart::TextArea::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    int left;
    int top;
    int right;
    int bottom;
    getFrameLeadings( left, top, right, bottom );
    return
        QRect( QPoint(0,0), areaGeometry().size() )
            .adjusted( left, top, -right, -bottom );
}
```

7.18.4.20 bool CartesianAxis::isAbscissa () const `[virtual]`

Definition at line 164 of file KDChartCartesianAxis.cpp.

References Bottom, position(), and Top.

Referenced by paintCtx(), and tickLength().

```
{
    return position() == Bottom || position() == Top;
}
```

7.18.4.21 bool CartesianAxis::isEmpty () const `[virtual]`

pure virtual in QLayoutItem

Definition at line 955 of file KDChartCartesianAxis.cpp.

```
{
    return false; // if the axis exists, it has some (perhaps default) content
}
```

7.18.4.22 bool CartesianAxis::isOrdinate () const `[virtual]`

Definition at line 169 of file KDChartCartesianAxis.cpp.

References Left, position(), and Right.

Referenced by paintCtx().

```
{
    return position() == Left || position() == Right;
}
```

7.18.4.23 QStringList AbstractAxis::labels () const `[inherited]`

Returns a list of strings, that are used as axis labels, as set via setLabels.

See also:
setLabels

Definition at line 273 of file KDChartAbstractAxis.cpp.

References d.

Referenced by maximumSize(), and paintCtx().

```
{
    return d->hardLabels;
}
```

7.18.4.24 void CartesianAxis::layoutPlanes () `[virtual]`

Definition at line 150 of file KDChartCartesianAxis.cpp.

References d, and KDChart::AbstractCoordinatePlane::layoutPlanes().

Referenced by resetTitleTextAttributes(), setPosition(), setTitleText(), and setTitleTextAttributes().

```
151 {
152     //qDebug() << "CartesianAxis::layoutPlanes()";
153     if( ! d->diagram() || ! d->diagram()->coordinatePlane() ) {
154         //qDebug() << "CartesianAxis::layoutPlanes(): Sorry, found no plane.";
155         return;
156     }
157     AbstractCoordinatePlane* plane = d->diagram()->coordinatePlane();
158     if( plane ){
159         plane->layoutPlanes();
160         //qDebug() << "CartesianAxis::layoutPlanes() OK";
161     }
162 }
```

7.18.4.25 int AbstractArea::leftOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the left edge of the area.

Note:

The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 77 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
78 {
79     // Re-calculate the sizes,
80     // so we also get the amountOf..Overlap members set newly:
81     if( ! doNotRecalculate )
82         sizeHint();
83     return d->amountOfLeftOverlap;
84 }
```

7.18.4.26 QSize CartesianAxis::maximumSize () const `[virtual]`

pure virtual in QLayoutItem

Definition at line 1007 of file KDChartCartesianAxis.cpp.

References Bottom, calculateOverlap(), KDChart::AbstractAxis::customizedLabel(), d, KDChart::AbstractCoordinatePlane::gridDimensionsList(), KDChart::TextAttributes::isVisible(), KDChart::AbstractAxis::labels(), Left, KDChart::AbstractCoordinatePlane::parent(), position(), KDChart::TextLayoutItem::realFont(), referenceDiagramIsBarDiagram(), Right, KDChart::TextLayoutItem::setText(), KDChart::TextLayoutItem::sizeHint(), KDChart::AbstractAxis::textAttributes(), tickLength(), titleText(), and Top.

Referenced by minimumSize(), and sizeHint().

```
1008 {
1009     QSize result;
1010     if ( !d->diagram() )
```

```
        return result;

    const TextAttributes labelTA = textAttributes();
    const bool drawLabels = labelTA.isVisible();

    const TextAttributes titleTA( d->titleTextAttributesWithAdjustedRotation() );
    const bool drawTitle = titleTA.isVisible() && ! titleText().isEmpty();

    AbstractCoordinatePlane* plane = d->diagram()->coordinatePlane();
    //qDebug() << this<<"::maximumSize() uses plane geometry" << plane->geometry();
    QObject* refArea = plane->parent();
    TextLayoutItem labelItem( QString::null, labelTA, refArea,
                              KDChartEnums::MeasureOrientationMinimum, Qt::AlignLeft );
    TextLayoutItem titleItem( titleText(), titleTA, refArea,
                              KDChartEnums::MeasureOrientationMinimum, Qt::AlignHCenter | Qt::AlignVC
    const qreal labelGap =
        drawLabels
        ? (QFontMetricsF( labelItem.realFont() ).height() / 3.0)
        : 0.0;
    const qreal titleGap =
        drawTitle
        ? (QFontMetricsF( titleItem.realFont() ).height() / 3.0)
        : 0.0;

    switch ( position() )
    {
    case Bottom:
    case Top: {
        const bool isBarDiagram = referenceDiagramIsBarDiagram(d->diagram());
        int leftOverlap = 0;
        int rightOverlap = 0;

        qreal w = 10.0;
        qreal h = 0.0;
        if( drawLabels ){
            // if there're no label strings, we take the biggest needed number as height
            if ( labels().count() ){
                // find the longest label text:
                const int first=0;
                const int last=labels().count()-1;
                for ( int i = first; i <= last; ++i )
                {
                    labelItem.setText( customizedLabel(labels()[ i ]) );
                    const QSize siz = labelItem.sizeHint();
                    h = qMax( h, static_cast<qreal>(siz.height()) );
                    calculateOverlap( i, first, last, siz.width(), isBarDiagram,
                                      leftOverlap, rightOverlap );

                }
            }else{
                QStringList headerLabels = d->diagram()->itemRowLabels();
                const int headerLabelsCount = headerLabels.count();
                if( headerLabelsCount ){
                    const int first=0;
                    const int last=headerLabelsCount-1;
                    for ( int i = first; i <= last; ++i )
                    {
                        labelItem.setText( customizedLabel(headerLabels[ i ]) );
                        const QSize siz = labelItem.sizeHint();
                        h = qMax( h, static_cast<qreal>(siz.height()) );
                        calculateOverlap( i, first, last, siz.width(), isBarDiagram,
                                          leftOverlap, rightOverlap );
                    }
                }else{
                    labelItem.setText(
                            customizedLabel(
                                    QString::number( plane->gridDimensionsList().first().end, 'f', 0
                    const QSize siz = labelItem.sizeHint();
                    h = siz.height();
                    calculateOverlap( 0, 0, 0, siz.width(), isBarDiagram,
                                      leftOverlap, rightOverlap );
                }
            }
            // we leave a little gap between axis labels and bottom (or top, resp.) side of axis
            h += labelGap;
        }
        // space for a possible title:
        if ( drawTitle ) {
            // we add the title height and leave a little gap between axis labels and axis title
            h += titleItem.sizeHint().height() + titleGap;
            w = titleItem.sizeHint().width() + 2.0;
        }
        // space for the ticks
        h += qAbs( tickLength() ) * 3.0;
        result = QSize ( static_cast<int>( w ), static_cast<int>( h ) );


        // If necessary adjust the widths
        // of the left (or right, resp.) side neighboring columns:
        d->amountOfLeftOverlap = leftOverlap;
        d->amountOfRightOverlap = rightOverlap;
        /* Unused code for a push-model:
        if( leftOverlap || rightOverlap ){
            QTimer::singleShot(200, const_cast<CartesianAxis*>(this),
                               SLOT(adjustLeftRightGridColumnWidths()));
        }
        */
    }
        break;
    case Left:
    case Right: {
        int topOverlap = 0;
        int bottomOverlap = 0;

        qreal w = 0.0;
        qreal h = 10.0;
        if( drawLabels ){
            // if there’re no label strings, we take the biggest needed number as width
            if ( labels().count() == 0 )
            {
                labelItem.setText(
                        customizedLabel(
                                QString::number( plane->gridDimensionsList().last().end, ’f’, 0 )));
                const QSize siz = labelItem.sizeHint();
                w = siz.width();
                calculateOverlap( 0, 0, 0, siz.height(), false,// bar diagram flag is ignored for Ord
                                  topOverlap, bottomOverlap );
            }else{
                // find the longest label text:
                const int first=0;
                const int last=labels().count()-1;
                for ( int i = first; i <= last; ++i )
                {
                    labelItem.setText( customizedLabel(labels()[ i ]) );
                    const QSize siz = labelItem.sizeHint();
                    qreal lw = siz.width();
                    w = qMax( w, lw );
                    calculateOverlap( 0, 0, 0, siz.height(), false,// bar diagram flag is ignored for
                                      topOverlap, bottomOverlap );
                }
            }
            // we leave a little gap between axis labels and left (or right, resp.) side of axis
            w += labelGap;
        }
        // space for a possible title:
        if ( drawTitle ) {
            // we add the title height and leave a little gap between axis labels and axis title
            w += titleItem.sizeHint().width() + titleGap;
            h = titleItem.sizeHint().height() + 2.0;
            //qDebug() << "left/right axis title item size-hint:" << titleItem.sizeHint();
        }
        // space for the ticks
        w += qAbs( tickLength() ) * 3.0;

        result = QSize ( static_cast<int>( w ), static_cast<int>( h ) );
        //qDebug() << "left/right axis width:" << result << "   w:" << w;


        // If necessary adjust the heights
        // of the top (or bottom, resp.) side neighboring rows:
        d->amountOfTopOverlap = topOverlap;
        d->amountOfBottomOverlap = bottomOverlap;
        /* Unused code for a push-model:
        if( topOverlap || bottomOverlap ){
            QTimer::singleShot(200, const_cast<CartesianAxis*>(this),
                               SLOT(adjustTopBottomGridRowHeights()));
        }
        */
    }
        break;
    default:
        Q_ASSERT( false ); // all positions need to be handled
        break;
    };
//qDebug() << "*******************" << result;
    //result=QSize(0,0);
    return result;
}
```

7.18.4.27 QSize CartesianAxis::minimumSize () const `[virtual]`

pure virtual in QLayoutItem

Definition at line 1180 of file KDChartCartesianAxis.cpp.

References maximumSize().

```
{
    return maximumSize();
}
```

7.18.4.28 bool KDChart::AbstractAxis::observedBy (AbstractDiagram ∗ *diagram*) const `[inherited]`

Definition at line 324 of file KDChartAbstractAxis.cpp.

References d.

```
{
    return d->hasDiagram( diagram );
}
```

7.18.4.29 void CartesianAxis::paint (QPainter ∗) `[virtual]`

reimpl

Implements KDChart::AbstractLayoutItem.

Definition at line 193 of file KDChartCartesianAxis.cpp.

References KDChart::AbstractArea::areaGeometry(), d, paintCtx(), KDChart::PaintContext::setCoordinatePlane(), KDChart::PaintContext::setPainter(), and KDChart::PaintContext::setRectangle().

```
{
    if( ! d->diagram() || ! d->diagram()->coordinatePlane() ) return;
    PaintContext ctx;
    ctx.setPainter ( painter );
    ctx.setCoordinatePlane( d->diagram()->coordinatePlane() );
    const QRect rect( areaGeometry() );

    //qDebug() << "CartesianAxis::paint( QPainter* painter )  " << " areaGeometry()():" << rect << " s

    ctx.setRectangle(
        QRectF (
            //QPointF(0, 0),
            QPointF(rect.left(), rect.top()),
            QSizeF(rect.width(), rect.height() ) ) );
    // enabling clipping so that we're not drawing outside
    QRegion clipRegion( rect.adjusted( -1, -1, 1, 1 ) );
    painter->save();
    painter->setClipRegion( clipRegion );
    paintCtx( &ctx );
    painter->restore();
    //qDebug() << "KDChart::CartesianAxis::paint() done.";
}
```

7.18.4.30 void AbstractArea::paintAll (QPainter & *painter*) `[virtual, inherited]`

Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

Reimplemented from KDChart::AbstractLayoutItem.

Definition at line 123 of file KDChartAbstractArea.cpp.

References KDChart::AbstractArea::areaGeometry(), d, KDChart::AbstractAreaBase::innerRect(), KDChart::AbstractLayoutItem::paint(), KDChart::AbstractAreaBase::paintBackground(), and KDChart::AbstractAreaBase::paintFrame().

Referenced by KDChart::AbstractArea::paintIntoRect().

```
{
    // Paint the background and frame
    const QRect overlappingArea( geometry().adjusted(
            -d->amountOfLeftOverlap,
            -d->amountOfTopOverlap,
            d->amountOfRightOverlap,
            d->amountOfBottomOverlap ) );
    paintBackground( painter, overlappingArea );
    paintFrame(      painter, overlappingArea );

    // temporarily adjust the widget size, to be sure all content gets calculated
    // to fit into the inner rectangle
    const QRect oldGeometry( areaGeometry()  );
    QRect inner( innerRect() );
    inner.moveTo(
        oldGeometry.left() + inner.left(),
        oldGeometry.top()  + inner.top() );
    const bool needAdjustGeometry = oldGeometry != inner;
    if( needAdjustGeometry )
        setGeometry( inner );
    paint( &painter );
    if( needAdjustGeometry )
        setGeometry( oldGeometry );
    //qDebug() << "AbstractAreaWidget::paintAll() done.";
}
```

7.18.4.31 void AbstractAreaBase::paintBackground (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 188 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintBackgroundAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintBackground()",
                "Private class was not initialized!" );
    paintBackgroundAttributes( painter, rect, d->backgroundAttributes );
}
```

7.18.4.32 void AbstractAreaBase::paintBackgroundAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::BackgroundAttributes & *attributes*) `[static, inherited]`

Definition at line 119 of file KDChartAbstractAreaBase.cpp.

References KDChart::BackgroundAttributes::brush(), KDChart::BackgroundAttributes::isVisible(), KDChart::BackgroundAttributes::pixmap(), and KDChart::BackgroundAttributes::pixmapMode().

Referenced by KDChart::AbstractAreaBase::paintBackground().

```
{
    if( !attributes.isVisible() ) return;

    /* first draw the brush (may contain a pixmap)*/
    if( Qt::NoBrush != attributes.brush().style() ) {
        KDChart::PainterSaver painterSaver( &painter );
        painter.setPen( Qt::NoPen );
        const QPointF newTopLeft( painter.deviceMatrix().map( rect.topLeft() ) );
        painter.setBrushOrigin( newTopLeft );
        painter.setBrush( attributes.brush() );
        painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
    }
    /* next draw the backPixmap over the brush */
    if( !attributes.pixmap().isNull() &&
        attributes.pixmapMode() != BackgroundAttributes::BackgroundPixmapModeNone ) {
        QPointF ol = rect.topLeft();
        if( BackgroundAttributes::BackgroundPixmapModeCentered == attributes.pixmapMode() )
        {
            ol.setX( rect.center().x() - attributes.pixmap().width() / 2 );
            ol.setY( rect.center().y() - attributes.pixmap().height()/ 2 );
            painter.drawPixmap( ol, attributes.pixmap() );
        } else {
            QMatrix m;
            double zW = (double)rect.width()  / (double)attributes.pixmap().width();
            double zH = (double)rect.height() / (double)attributes.pixmap().height();
            switch( attributes.pixmapMode() ) {
            case BackgroundAttributes::BackgroundPixmapModeScaled:
            {
                double z;
                z = qMin( zW, zH );
                m.scale( z, z );
            }
            break;
            case BackgroundAttributes::BackgroundPixmapModeStretched:
                m.scale( zW, zH );
                break;
            default:
                ; // Cannot happen, previously checked
            }
            QPixmap pm = attributes.pixmap().transformed( m );
            ol.setX( rect.center().x() - pm.width() / 2 );
            ol.setY( rect.center().y() - pm.height()/ 2 );
            painter.drawPixmap( ol, pm );
        }
    }
}
```

7.18.4.33 void CartesianAxis::paintCtx (PaintContext ∗) `[virtual]`

reimpl

Reimplemented from KDChart::AbstractLayoutItem.

Definition at line 370 of file KDChartCartesianAxis.cpp.

References KDChart::AbstractArea::areaGeometry(), Bottom, KDChart::DataDimension::calcMode, calculateNextLabel(), KDChart::PaintContext::coordinatePlane(), KDChart::AbstractAxis::customizedLabel(), d, KDChart::DataDimensionsList, KDChart::DataDimension::distance(), KDChart::DataDimension::end, KDChart::TextLayoutItem::geometry(), geometry(), KDChart::AbstractCoordinatePlane::gridDimensionsList(), KDChart::TextLayoutItem::intersects(), isAbscissa(), KDChart::DataDimension::isCalculated, isOrdinate(), KDChart::TextAttributes::isVisible(), KDChart::AbstractAxis::labels(), Left, KDChart::TextLayoutItem::paint(), KDChart::PaintContext::painter(), KDChart::AbstractCoordinatePlane::parent(), position(), KDChart::TextLayoutItem::realFont(), referenceDiagramIsBarDiagram(), Right, KDChart::TextLayoutItem::setGeometry(), KDChart::TextLayoutItem::setText(), KDChart::AbstractAxis::shortLabels(), KDChart::TextLayoutItem::sizeHint(), KDChart::DataDimension::start, KDChart::DataDimension::stepWidth, KDChart::DataDimension::subStepWidth, KDChart::TextLayoutItem::text(), KDChart::AbstractAxis::textAttributes(), tickLength(), titleText(), Top, and KDChart::CartesianCoordinatePlane::translate().

Referenced by paint().

```
{

    Q_ASSERT_X ( d->diagram(), "CartesianAxis::paint",
                 "Function call not allowed: The axis is not assigned to any diagram." );

    CartesianCoordinatePlane* plane = dynamic_cast<CartesianCoordinatePlane*>(context->coordinatePlane
    Q_ASSERT_X ( plane, "CartesianAxis::paint",
                 "Bad function call: PaintContext::coodinatePlane() NOT a cartesian plane." );

    // note: Not having any data model assigned is no bug
    //       but we can not draw an axis then either.
    if( ! d->diagram()->model() )
```

```
        return;


    /*
     * let us paint the labels at a
     * smaller resolution
     * Same mini pixel value as for
     * Cartesian Grid
     */
    //const qreal MinimumPixelsBetweenRulers = 1.0;
    DataDimensionsList dimensions( plane->gridDimensionsList() );
    //qDebug("CartesianAxis::paintCtx() gets DataDimensionsList.first():   start: %f   end: %f   stepW

    // test for programming errors: critical
    Q_ASSERT_X ( dimensions.count() == 2, "CartesianAxis::paint",
                 "Error: plane->gridDimensionsList() did not return exactly two dimensions." );
    DataDimension dimX =
            AbstractGrid::adjustedLowerUpperRange( dimensions.first(), true, true );
    const DataDimension dimY =
            AbstractGrid::adjustedLowerUpperRange( dimensions.last(), true, true );
    const DataDimension& dim = (isAbscissa() ? dimX : dimY);

    /*
    if(isAbscissa())
        qDebug() << "         " << "Abscissa:" << dimX.start <<".."<<dimX.end <<"  step"<<dimX.stepWi
    else
        qDebug() << "         " << "Ordinate:" << dimY.start <<".."<<dimY.end <<"  step"<<dimY.stepWi
    */


    /*
     * let us paint the labels at a
     * smaller resolution
     * Same mini pixel value as for
     * Cartesian Grid
     */
    const qreal MinimumPixelsBetweenRulers = qMin(  dimX.stepWidth,  dimY.stepWidth );//1.0;

    // preparations:
    // - calculate the range that will be displayed:
    const qreal absRange = qAbs( dim.distance() );

    qreal numberOfUnitRulers;
    if ( isAbscissa() ) {
        if( dimX.isCalculated )
            numberOfUnitRulers = absRange / qAbs( dimX.stepWidth ) + 1.0;
        else
            numberOfUnitRulers = d->diagram()->model()->rowCount() - 1.0;
    }else{
        numberOfUnitRulers = absRange / qAbs( dimY.stepWidth ) + 1.0;

    }

    //    qDebug() << "absRange" << absRange << "dimY.stepWidth:" << dimY.stepWidth << "numberOfUnitRu

    qreal numberOfSubUnitRulers;
    if ( isAbscissa() ){
        if( dimX.isCalculated )
            numberOfSubUnitRulers = absRange / qAbs( dimX.subStepWidth ) + 1.0;
        else
            numberOfSubUnitRulers = dimX.subStepWidth>0 ? absRange / qAbs( dimX.subStepWidth ) + 1.0 :
    }else{
        numberOfSubUnitRulers = absRange / qAbs( dimY.subStepWidth ) + 1.0;
    }

    // - calculate the absolute range in screen pixels:
    const QPointF p1 = plane->translate( QPointF(dimX.start, dimY.start) );
    const QPointF p2 = plane->translate( QPointF(dimX.end,   dimY.end) );

    double screenRange;
    if ( isAbscissa() )
    {
        screenRange = qAbs ( p1.x() - p2.x() );
    } else {
        screenRange = qAbs ( p1.y() - p2.y() );
    }

    const bool useItemCountLabels = isAbscissa() && ! dimX.isCalculated;
    //qDebug() << "CartesianAxis::paintCtx useItemCountLabels "<< useItemCountLabels;

    //qDebug() << "isAbscissa():" << isAbscissa() << "   dimX.isCalculated:" << dimX.isCalculated << "
    //FIXME(khz): Remove this code, and do the calculation in the grid calc function
    if( isAbscissa() && ! dimX.isCalculated ){
        // dont ignore the users settings
        dimX.stepWidth = dimX.stepWidth ? dimX.stepWidth : 1.0;
        //qDebug() << "screenRange / numberOfUnitRulers <= MinimumPixelsBetweenRulers" << screenRange
        while( screenRange / numberOfUnitRulers <= MinimumPixelsBetweenRulers ){
            dimX.stepWidth *= 10.0;
            dimX.subStepWidth  *= 10.0;
            numberOfUnitRulers = qAbs( dimX.distance() / dimX.stepWidth );
        }
    }

    const bool drawUnitRulers = screenRange / ( numberOfUnitRulers / dimX.stepWidth ) > MinimumPixelsB
    const bool drawSubUnitRulers =
        (numberOfSubUnitRulers != 0.0) &&
        (screenRange / numberOfSubUnitRulers > MinimumPixelsBetweenRulers);

    const TextAttributes labelTA = textAttributes();
    const bool drawLabels = labelTA.isVisible();

    // - find the reference point at which to start drawing and the increment (line distance);
    QPointF rulerRef;
    const QRect areaGeoRect( areaGeometry() );
    const QRect geoRect( geometry() );
    QRectF rulerRect;
    double rulerWidth;
    double rulerHeight;

    QPainter* ptr = context->painter();

    //for debugging: if( isAbscissa() )ptr->drawRect(areaGeoRect.adjusted(0,0,-1,-1));
    //qDebug() << "         " << (isAbscissa() ? "Abscissa":"Ordinate") << "axis painting with geometr

    // FIXME references are of course different for all locations:
    rulerWidth = areaGeoRect.width();
    rulerHeight =  areaGeoRect.height();
    switch( position() )
    {
    case Top:
        rulerRef.setX( areaGeoRect.topLeft().x() );
        rulerRef.setY( areaGeoRect.topLeft().y() + rulerHeight );
        break;
    case Bottom:
        rulerRef.setX( areaGeoRect.bottomLeft().x() );
        rulerRef.setY( areaGeoRect.bottomLeft().y() - rulerHeight );
        break;
    case Right:
        rulerRef.setX( areaGeoRect.bottomRight().x() - rulerWidth );
        rulerRef.setY( areaGeoRect.bottomRight().y() );
        break;
    case Left:
        rulerRef.setX( areaGeoRect.bottomLeft().x() + rulerWidth );
        rulerRef.setY( areaGeoRect.bottomLeft().y() );
        break;
    }

    // set up the lines to paint:

    // set up a map of integer positions,

    // - starting with the fourth
    // - the the halfs
    // - then the tens
    // this will override all halfs and fourth that hit a higher-order ruler
    // MAKE SURE TO START AT (0, 0)!

    // set up a reference point,  a step vector and a unit vector for the drawing:

    const qreal minValueY = dimY.start;
    const qreal maxValueY = dimY.end;
    const qreal minValueX = dimX.start;
    const qreal maxValueX = dimX.end;
    const bool isLogarithmicX = (dimX.calcMode == AbstractCoordinatePlane::Logarithmic );
    const bool isLogarithmicY = (dimY.calcMode == AbstractCoordinatePlane::Logarithmic );
//#define AXES_PAINTING_DEBUG 1
#ifdef AXES_PAINTING_DEBUG
    qDebug() << "CartesianAxis::paint: reference values:" << endl
             << "-- range x/y: " << dimX.distance() << "/" << dimY.distance() << endl
             << "-- absRange: " << absRange << endl
             << "-- numberOfUnitRulers: " << numberOfUnitRulers << endl
             << "-- screenRange: " << screenRange << endl
             << "-- drawUnitRulers: " << drawUnitRulers << endl
             << "-- drawLabels: " << drawLabels << endl
             << "-- ruler reference point:: " << rulerRef << endl
             << "-- minValueX: " << minValueX << "   maxValueX: " << maxValueX << endl
             << "-- minValueY: " << minValueY << "   maxValueY: " << maxValueY << endl
        ;
#endif

    ptr->setPen ( Qt::black );

    const QObject* referenceArea = plane->parent();

    // that QVector contains all drawn x-ticks (so no subticks are drawn there also)
    QVector< int > drawnXTicks;
    // and that does the same for the y-ticks
    QVector< int > drawnYTicks;

    /*
     * Find out if it is a bar diagram
     * bar diagrams display their data per column
     * we need to handle the last label another way
     * 1 - Last label == QString null ( Header Labels )
     * 2 - Display labels and ticks in the middle of the column
     */

    const bool isBarDiagram = referenceDiagramIsBarDiagram(d->diagram());

    // this draws the unit rulers
    if ( drawUnitRulers ) {
        const int hardLabelsCount  = labels().count();
        const int shortLabelsCount = shortLabels().count();
        bool useShortLabels = false;


        bool useConfiguredStepsLabels = false;
        QStringList headerLabels;
        if( useItemCountLabels ){
            //qDebug() << (isOrdinate() ? "is Ordinate" : "is Abscissa");
            headerLabels =
                isOrdinate()
                ? d->diagram()->datasetLabels()
                : d->diagram()->itemRowLabels();
            // check if configured stepWidth
            useConfiguredStepsLabels = isAbscissa() &&
                    dimX.stepWidth &&
                    (( (headerLabels.count() - 1)/ dimX.stepWidth ) != numberOfUnitRulers);
            if( useConfiguredStepsLabels ) {
                numberOfUnitRulers = ( headerLabels.count() - 1 )/ dimX.stepWidth;
                // we need to register data values for the steps
                // in case it is configured by the user
                QStringList configuredStepsLabels;
                double value = headerLabels.first().toDouble();
                configuredStepsLabels << QString::number( value );
                for (  int i = 0; i < numberOfUnitRulers; i++ ) {
                    value += dimX.stepWidth;
                    configuredStepsLabels.append( QString::number( value ) );
                }
                headerLabels = configuredStepsLabels;
            }

            if (  isBarDiagram )
                headerLabels.append( QString::null );
        }


        const int headerLabelsCount = headerLabels.count();
        //qDebug() << "headerLabelsCount" << headerLabelsCount;

        TextLayoutItem* labelItem =
            drawLabels
            ? new TextLayoutItem( QString::number( minValueY ),
                                  labelTA,
                                  referenceArea,
                                  KDChartEnums::MeasureOrientationMinimum,
                                  Qt::AlignLeft )
            : 0;
        TextLayoutItem* labelItem2 =
            drawLabels
            ? new TextLayoutItem( QString::number( minValueY ),
                                  labelTA,
                                  referenceArea,
                                  KDChartEnums::MeasureOrientationMinimum,
                                  Qt::AlignLeft )
            : 0;
        const QFontMetricsF met(
            drawLabels
            ? labelItem->realFont()
            : QFontMetricsF( QApplication::font() ) );
        const qreal halfFontHeight = met.height() * 0.5;

        if ( isAbscissa() ) {
            // If we have a labels list AND a short labels list, we first find out,
            // if there is enough space for the labels: if not, use the short labels.
            if( drawLabels && hardLabelsCount > 0 && shortLabelsCount > 0 ){
                bool labelsAreOverlapping = false;
                int iLabel = 0;
                qreal i = minValueX;
                while ( i < maxValueX && !labelsAreOverlapping )
                {
                    if ( dimX.stepWidth != 1.0 && ! dim.isCalculated )
                    {
                        labelItem->setText(  customizedLabel(QString::number( i, ’f’, 0 )) );
                        labelItem2->setText( customizedLabel(QString::number( i + dimX.stepWidth, ’f’,
                    } else {

                        int index = iLabel;
                        labelItem->setText(  customizedLabel(labels()[ index < hardLabelsCount ? index
                        labelItem2->setText( customizedLabel(labels()[ index < hardLabelsCount - 1 ? i
                    }
                    QPointF firstPos( i, 0.0 );
                    firstPos = plane->translate( firstPos );

                    QPointF secondPos( i + dimX.stepWidth, 0.0 );
                    secondPos = plane->translate( secondPos );

                    labelsAreOverlapping = labelItem->intersects( *labelItem2, firstPos, secondPos );
                    if ( iLabel++ > hardLabelsCount - 1 )
                        iLabel = 0;
                    if ( isLogarithmicX )
                        i *= 10.0;
                    else
                        i += dimX.stepWidth;

                }

                useShortLabels = labelsAreOverlapping;
            }

            qreal labelDiff = dimX.stepWidth;
            //qDebug() << "labelDiff " << labelDiff;
            if ( drawLabels )
            {

                qreal i = minValueX;
                int iLabel = 0;
                const int precision = ( QString::number( labelDiff  ).section( QLatin1Char(’.’), 1,  2

                while ( i + labelDiff < maxValueX )
                {

                    //qDebug() << "drawLabels" << drawLabels << "  hardLabelsCount" << hardLabelsCount
                    //        << "  dimX.stepWidth" << dimX.stepWidth << "  dim.isCalculated" << dim.i
                    if ( !drawLabels || hardLabelsCount < 1 || ( dimX.stepWidth != 1.0 && ! dim.isCalc
                    {
                        // Check intersects for the header label - we need to pass the full string
                        // here and not only the i value.
                        if( useConfiguredStepsLabels ){
                            labelItem->setText( customizedLabel(headerLabels[ iLabel   ]) );
                            labelItem2->setText(customizedLabel(headerLabels[ iLabel+1 ]) );
                        }else{
                            //qDebug() << "i + labelDiff " << i + labelDiff;
                            labelItem->setText( customizedLabel(headerLabelsCount ? headerLabels[stati
                                : QString::number( i, ’f’, precision )) );
                            //           qDebug() << "1 - labelItem->text() " << labelItem->text();
                            //qDebug() << "labelDiff" << labelDiff
                            //        << "  index" << i+labelDiff << "  count" << headerLabelsCount;
                            labelItem2->setText( customizedLabel(headerLabelsCount ? headerLabels[stat
                                : QString::number( i + labelDiff, ’f’, precision )) );
                            //qDebug() << "2 - labelItem->text() " << labelItem->text();
                            //qDebug() << "labelItem2->text() " << labelItem2->text();
                        }
                    } else {
                        const int idx = (iLabel < hardLabelsCount    ) ? iLabel     : 0;
                        const int idx2= (iLabel < hardLabelsCount - 1) ? iLabel + 1 : 0;
                        labelItem->setText(  customizedLabel(
                                useShortLabels ? shortLabels()[ idx ] : labels()[ idx ]) );
                        labelItem2->setText( customizedLabel(
                                useShortLabels ? shortLabels()[ idx2] : labels()[ idx2]) );
                    }

                    QPointF firstPos( i, 0.0 );
                    firstPos = plane->translate( firstPos );

                    QPointF secondPos( i + labelDiff, 0.0 );
                    secondPos = plane->translate( secondPos );


                    if ( labelItem->intersects( *labelItem2, firstPos, secondPos ) )
                    {
                        i = minValueX;
                        labelDiff += labelDiff;
                        iLabel = 0;
                    }
                    else
                    {
                        i += labelDiff;
                    }

                    ++iLabel;
                    if ( (iLabel > hardLabelsCount - 1) && !useConfiguredStepsLabels )
                    {
                        iLabel = 0;
                    }
                }
            }

            int idxLabel = 0;
            qreal iLabelF = minValueX;
            qreal i = minValueX;
            qreal labelStep = 0.0;
            //qDebug() << "dimX.stepWidth:" << dimX.stepWidth;
            //dimX.stepWidth = 0.5;
            while( i <= maxValueX ) {
                // Line charts: we want the first tick to begin at 0.0 not at 0.5 otherwise labels and
                // values does not fit each others
                QPointF topPoint ( i + ( isBarDiagram ? 0.5 : 0.0 ), 0.0 );
                QPointF bottomPoint ( topPoint );
                topPoint = plane->translate( topPoint );
                bottomPoint = plane->translate( bottomPoint );
                topPoint.setY( rulerRef.y() + tickLength() );
                bottomPoint.setY( rulerRef.y() );

                const qreal translatedValue = topPoint.x();
                const bool bIsVisibleLabel =
                        ( translatedValue >= geoRect.left() && translatedValue <= geoRect.right() );

                //Dont paint more ticks than we need
                //when diagram type is Bar
                bool painttick = true;
                if (  isBarDiagram && i == maxValueX )
                    painttick = false;

                if ( bIsVisibleLabel && painttick )
                    ptr->drawLine( topPoint, bottomPoint );

                drawnXTicks.append( static_cast<int>( topPoint.x() ) );
                if( drawLabels ) {
                    if( bIsVisibleLabel ){
                        if ( isLogarithmicX )
                            labelItem->setText( customizedLabel(QString::number( i, ’f’, 0 )) );
                        /* We dont need that
                        * it causes header labels to be skipped even if there is enough
                        * space for them to displayed.
                        * Commenting for now - I need to test more in details - Let me know if I am wr
                        */
                        /*
                        else if( (dimX.stepWidth != 1.0) && ! dimX.isCalculated ) {
                        labelItem->setText( customizedLabel(QString::number( i, ’f’, 0 )) );
                        }
                        */
```

```
                        else {
                            labelItem->setText(
                                    customizedLabel(
                                          hardLabelsCount
                                        ? ( useShortLabels    ? shortLabels()[ idxLabel ] : labels()[
                                        : ( headerLabelsCount ? headerLabels[  idxLabel ] : QString::
                        }
                        // No need to call labelItem->setParentWidget(), since we are using
                        // the layout item temporarily only.
                        if( labelStep <= 0 ) {
                            const QSize size( labelItem->sizeHint() );
                            labelItem->setGeometry(
                                QRect(
                                    QPoint(
                                        static_cast<int>( topPoint.x() - size.width() / 2 ),
                                        static_cast<int>( topPoint.y() +
                                                        ( position() == Bottom
                                                            ? halfFontHeight
                                                            : ((halfFontHeight + size.height()) * -1.
                                    size ) );

                            bool origClipping = ptr->hasClipping();

                            QRect labelGeo = labelItem->geometry();
                            // if our item would only half fit, we disable clipping for that one
                            if( labelGeo.left() < geoRect.left() && labelGeo.right() > geoRect.left()
                                ptr->setClipping( false );
                            if( labelGeo.left() < geoRect.right() && labelGeo.right() > geoRect.right(
                                ptr->setClipping( false );

                            labelItem->setGeometry( labelGeo );

                            labelStep = labelDiff - dimX.stepWidth;
                            labelItem->paint( ptr );

                            // do not call customizedLabel() again:
                            labelItem2->setText( labelItem->text() );

                            // maybe enable clipping afterwards
                            ptr->setClipping( origClipping );
                        } else {
                            labelStep -= dimX.stepWidth;
                        }
                    }

                    if( hardLabelsCount ) {
                        if( idxLabel >= hardLabelsCount  -1 )
                            idxLabel = 0;
                        else
                            ++idxLabel;
                    } else if( headerLabelsCount ) {
                        if( idxLabel >= headerLabelsCount - 1 ) {
                            idxLabel = 0;
                        }else
                            ++idxLabel;
                    } else {
                        iLabelF += dimX.stepWidth;
                    }
                }
                if ( isLogarithmicX )
                    i *= 10.0;
                else
                    i += dimX.stepWidth;
            }
        } else {
            const double maxLimit = maxValueY;
            const double steg = dimY.stepWidth;
            int maxLabelsWidth = 0;
            qreal labelValue;
            if( drawLabels && position() == Right ){
                // Find the widest label, so we to know how much we need to right-shift
                // our labels, to get them drawn right aligned:
                labelValue = minValueY;
                while ( labelValue <= maxLimit ) {
                    labelItem->setText( customizedLabel(QString::number( labelValue )) );
                    maxLabelsWidth = qMax( maxLabelsWidth, labelItem->sizeHint().width() );

                    calculateNextLabel( labelValue, steg, isLogarithmicY );
                }
            }

            bool origClipping = ptr->hasClipping();
            ptr->setClipping( false );
            labelValue = minValueY;
            qreal step = steg;
            bool nextLabel = false;
            //qDebug("minValueY: %f   maxLimit: %f   steg: %f", minValueY, maxLimit, steg);

            // first calculate the steps depending on labels colision
            while ( labelValue <= maxLimit ) {
                QPointF leftPoint = plane->translate( QPointF( 0, labelValue ) );
                const qreal translatedValue = leftPoint.y();
                //qDebug() << "geoRect:" << geoRect << "   geoRect.top()" << geoRect.top()
                //<< "geoRect.bottom()" << geoRect.bottom() << "  translatedValue:" << translatedValue
                if( translatedValue > geoRect.top() && translatedValue <= geoRect.bottom() ){
                    if ( drawLabels ) {
                        labelItem->setText(  customizedLabel(QString::number( labelValue )) );
                        labelItem2->setText( customizedLabel(QString::number( labelValue + step )) );
                        QPointF nextPoint = plane->translate(  QPointF( 0,  labelValue + step ) );
                        if ( labelItem->intersects( *labelItem2, leftPoint, nextPoint ) )
                        {
                            step += steg;
                            nextLabel = false;
                        }else{
                            nextLabel = true;
                        }
                    }
                }else{
                    nextLabel = true;
                }

                if ( nextLabel || isLogarithmicY )
                    calculateNextLabel( labelValue, step, isLogarithmicY );
                else
                    labelValue = minValueY;
            }

            // Second - Paint the labels
            labelValue = minValueY;
            //qDebug() << "axis labels starting at" << labelValue << "step width" << step;
            while ( labelValue <= maxLimit ) {
                //qDebug() << "value now" << labelValue;
                labelItem->setText( customizedLabel(QString::number( labelValue )) );
                QPointF leftPoint = plane->translate( QPointF( 0, labelValue ) );
                QPointF rightPoint ( 0.0, labelValue );
                rightPoint = plane->translate( rightPoint );
                leftPoint.setX( rulerRef.x() + tickLength() );
                rightPoint.setX( rulerRef.x() );

                const qreal translatedValue = rightPoint.y();
                const bool bIsVisibleLabel =
                        ( translatedValue >= geoRect.top() && translatedValue <= geoRect.bottom() );

                if( bIsVisibleLabel ){
                    ptr->drawLine( leftPoint, rightPoint );
                    drawnYTicks.append( static_cast<int>( leftPoint.y() ) );
                    const QSize labelSize( labelItem->sizeHint() );
                    leftPoint.setX( leftPoint.x() );
                    const int x =
                        static_cast<int>( leftPoint.x() + met.height() * ( position() == Left ? -0.5 :
                        - ( position() == Left ? labelSize.width() : (labelSize.width() - maxLabelsWi
                    const int y =
                        static_cast<int>( leftPoint.y() - ( met.ascent() + met.descent() ) * 0.6 );
                    labelItem->setGeometry( QRect( QPoint( x, y ), labelSize ) );
                    labelItem->paint( ptr );
                }

                calculateNextLabel( labelValue, step, isLogarithmicY );
            }

            ptr->setClipping( origClipping );
        }
        delete labelItem;
        delete labelItem2;
    }

    // this draws the subunit rulers
    if ( drawSubUnitRulers ) {
        d->drawSubUnitRulers( ptr, plane, dim, rulerRef, isAbscissa() ? drawnXTicks : drawnYTicks );
    }

    if( ! titleText().isEmpty() ){
        d->drawTitleText( ptr, plane, areaGeoRect );
    }

    //qDebug() << "KDChart::CartesianAxis::paintCtx() done.";
}
```

7.18.4.34 void AbstractAreaBase::paintFrame (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 196 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintFrameAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintFrame()",
                "Private class was not initialized!" );
    paintFrameAttributes( painter, rect, d->frameAttributes );
}
```

7.18.4.35 void AbstractAreaBase::paintFrameAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::FrameAttributes & *attributes*) `[static, inherited]`

Definition at line 169 of file KDChartAbstractAreaBase.cpp.

References KDChart::FrameAttributes::isVisible(), and KDChart::FrameAttributes::pen().

Referenced by KDChart::AbstractAreaBase::paintFrame().

```
{
```

```
172
173     if( !attributes.isVisible() ) return;
174
175     // Note: We set the brush to NoBrush explicitely here.
176     //       Otherwise we might get a filled rectangle, so any
177     //       previously drawn background would be overwritten by that area.
178
179     const QPen   oldPen(   painter.pen() );
180     const QBrush oldBrush( painter.brush() );
181     painter.setPen(   attributes.pen() );
182     painter.setBrush( Qt::NoBrush );
183     painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
184     painter.setBrush( oldBrush );
185     painter.setPen(   oldPen );
186 }
```

7.18.4.36 void AbstractArea::paintIntoRect (QPainter & *painter*, const QRect & *rect*) `[virtual, inherited]`

Draws the background and frame, then calls paint().

In most cases there is no need to overwrite this method in a derived class, but you would overwrite AbstractLayoutItem::paint() instead.

Definition at line 111 of file KDChartAbstractArea.cpp.

References KDChart::AbstractArea::paintAll().

```
112 {
113     const QRect oldGeometry( geometry() );
114     if( oldGeometry != rect )
115         setGeometry( rect );
116     painter.translate( rect.left(), rect.top() );
117     paintAll( painter );
118     painter.translate( -rect.left(), -rect.top() );
119     if( oldGeometry != rect )
120         setGeometry( oldGeometry );
121 }
```

7.18.4.37 QLayout∗ KDChart::AbstractLayoutItem::parentLayout () `[inherited]`

Definition at line 74 of file KDChartLayoutItems.h.

```
75           {
76               return mParentLayout;
77           }
```

7.18.4.38 const CartesianAxis::Position CartesianAxis::position () const `[virtual]`

Definition at line 145 of file KDChartCartesianAxis.cpp.

References d.

Referenced by expandingDirections(), isAbscissa(), isOrdinate(), maximumSize(), paintCtx(), and tickLength().

```
146 {
147     return d->position;
148 }
```

7.18.4.39 void AbstractArea::positionHasChanged () `[protected, virtual, inherited]`

Reimplemented from KDChart::AbstractAreaBase.

Definition at line 155 of file KDChartAbstractArea.cpp.

```
{
    emit positionChanged( this );
}
```

7.18.4.40 void KDChart::AbstractLayoutItem::removeFromParentLayout () `[inherited]`

Definition at line 78 of file KDChartLayoutItems.h.

Referenced by KDChart::Chart::takeCoordinatePlane().

```
        {
            if( mParentLayout ){
                if( widget() )
                    mParentLayout->removeWidget( widget() );
                else
                    mParentLayout->removeItem( this );
            }
        }
```

7.18.4.41 void CartesianAxis::resetTitleTextAttributes ()

Reset the title text attributes to the built-in default:.

Same font and pen as AbstractAxis::textAttributes() and 1.5 times their size.

Definition at line 127 of file KDChartCartesianAxis.cpp.

References d, and layoutPlanes().

```
{
    d->useDefaultTextAttributes = true;
    layoutPlanes();
}
```

7.18.4.42 int AbstractArea::rightOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the right edge of the area.

Note:

> The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 85 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
86 {
87     // Re-calculate the sizes,
88     // so we also get the amountOf..Overlap members set newly:
89     if( ! doNotRecalculate )
90         sizeHint();
91     return d->amountOfRightOverlap;
92 }
```

7.18.4.43 void AbstractAreaBase::setBackgroundAttributes (const BackgroundAttributes & *a*) `[inherited]`

Definition at line 107 of file KDChartAbstractAreaBase.cpp.

References d.

```
108 {
109     d->backgroundAttributes = a;
110 }
```

7.18.4.44 void AbstractAreaBase::setFrameAttributes (const FrameAttributes & *a*) `[inherited]`

Definition at line 97 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone().

```
98 {
99     d->frameAttributes = a;
100 }
```

7.18.4.45 void CartesianAxis::setGeometry (const QRect & *r*) `[virtual]`

pure virtual in QLayoutItem

Implements KDChart::AbstractAxis.

Definition at line 1190 of file KDChartCartesianAxis.cpp.

References d.

```
1191 {
1192 //    qDebug() << "KDChart::CartesianAxis::setGeometry(" << r << ") called"
1193 //             << (isAbscissa() ? "for Abscissa":"for Ordinate") << "axis";
1194     d->geometry = r;
1195 }
```

7.18.4.46 void AbstractAxis::setLabels (const QStringList & *list*) `[inherited]`

Use this to specify your own set of strings, to be used as axis labels.

Labels specified via setLabels take precedence: If a non-empty list is passed, KD Chart will use these strings as axis labels, instead of calculating them.

If you a smaller number of strings than the number of labels drawn at this axis, KD Chart will iterate over the list, repeating the strings, until all labels are drawn. As an example you could specify the seven days of the week as abscissa labels, which would be repeatedly used then.

By passing an empty QStringList you can reset the default behaviour.

See also:
labels, setShortLabels

Definition at line 263 of file KDChartAbstractAxis.cpp.

References d.

```
264 {
265     d->hardLabels = list;
266 }
```

7.18.4.47 void KDChart::AbstractLayoutItem::setParentLayout (QLayout ∗ *lay*) `[inherited]`

Definition at line 70 of file KDChartLayoutItems.h.

```
71          {
72              mParentLayout = lay;
73          }
```

7.18.4.48 void KDChart::AbstractLayoutItem::setParentWidget (QWidget ∗ *widget*) `[virtual, inherited]`

Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

Thus, you need to call setParentWidget on every item, that has a non-fixed size.

Definition at line 64 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::mParent.

Referenced by KDChart::Legend::buildLegend(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
65 {
66     mParent = widget;
67 }
```

7.18.4.49 void CartesianAxis::setPosition (Position *p*) `[virtual]`

Definition at line 139 of file KDChartCartesianAxis.cpp.

References d, and layoutPlanes().

```
140 {
141     d->position = p;
142     layoutPlanes();
143 }
```

7.18.4.50 void AbstractAxis::setShortLabels (const QStringList & *list*) `[inherited]`

Use this to specify your own set of strings, to be used as axis labels, in case the normal labels are too long.

Note:

Setting done via setShortLabels will be ignored, if you did not pass a non-empty string list via setLabels too!

By passing an empty QStringList you can reset the default behaviour.

See also:

shortLabels, setLabels

Definition at line 289 of file KDChartAbstractAxis.cpp.

References d.

```
{
    d->hardShortLabels = list;
}
```

7.18.4.51 void AbstractAxis::setTextAttributes (const TextAttributes & *a*) `[inherited]`

Use this to specify the text attributes to be used for axis labels.

By default, the reference area will be set at painting time. It will be the then-valid coordinate plane's parent widget, so normally, it will be the KDChart::Chart. Thus the labels of all of your axes in all of your diagrams within that Chart will be drawn in same font size, by default.

See also:

textAttributes, setLabels

Definition at line 231 of file KDChartAbstractAxis.cpp.

References d.

```
{
    d->textAttributes = a;
}
```

7.18.4.52 void CartesianAxis::setTitleText (const QString & *text*)

Definition at line 96 of file KDChartCartesianAxis.cpp.

References d, and layoutPlanes().

```
{
    //FIXME(khz): Call update al all places where axis internals are changed!
    d->titleText = text;
     layoutPlanes();
}
```

7.18.4.53 void CartesianAxis::setTitleTextAttributes (const TextAttributes & *a*)

Definition at line 108 of file KDChartCartesianAxis.cpp.

References d, and layoutPlanes().

```
{
    d->titleTextAttributes = a;
    d->useDefaultTextAttributes = false;
    layoutPlanes();
}
```

7.18.4.54 QStringList AbstractAxis::shortLabels () const `[inherited]`

Returns a list of strings, that are used as axis labels, as set via setShortLabels.

Note:

Setting done via setShortLabels will be ignored, if you did not pass a non-empty string list via setLabels too!

See also:

setShortLabels

Definition at line 302 of file KDChartAbstractAxis.cpp.

References d.

Referenced by paintCtx().

```
{
    return d->hardShortLabels;
}
```

7.18.4.55 QSize CartesianAxis::sizeHint () const `[virtual]`

pure virtual in QLayoutItem

Definition at line 1185 of file KDChartCartesianAxis.cpp.

References maximumSize().

```
{
    return maximumSize();
}
```

7.18.4.56 void KDChart::AbstractLayoutItem::sizeHintChanged () const `[virtual, inherited]`

Report changed size hint: ask the parent widget to recalculate the layout.

Definition at line 86 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextLayoutItem::sizeHint().

```
{
    // This is exactly like what QWidget::updateGeometry does.
//  qDebug("KDChart::AbstractLayoutItem::sizeHintChanged() called");
    if( mParent ) {
        if ( mParent->layout() )
            mParent->layout()->invalidate();
        else
            QApplication::postEvent( mParent, new QEvent( QEvent::LayoutRequest ) );
    }
}
```

7.18.4.57 TextAttributes AbstractAxis::textAttributes () const `[inherited]`

Returns the text attributes to be used for axis labels.

See also:
setTextAttributes

Definition at line 241 of file KDChartAbstractAxis.cpp.

References d.

Referenced by maximumSize(), paintCtx(), and titleTextAttributes().

```
{
    return d->textAttributes;
}
```

7.18.4.58 int CartesianAxis::tickLength (bool *subUnitTicks* = false) const

Definition at line 1202 of file KDChartCartesianAxis.cpp.

References isAbscissa(), Left, position(), and Top.

Referenced by maximumSize(), and paintCtx().

```
{
    int result = 0;

    if ( isAbscissa() ) {
        result = position() == Top ? -4 : 3;
    } else {
        result = position() == Left ? -4 : 3;
    }

    if ( subUnitTicks )
        result = result < 0 ? result + 1 : result - 1;

    return result;
}
```

7.18.4.59 QString CartesianAxis::titleText () const

Definition at line 103 of file KDChartCartesianAxis.cpp.

References d.

Referenced by maximumSize(), and paintCtx().

```
104 {
105     return d->titleText;
106 }
```

7.18.4.60 TextAttributes CartesianAxis::titleTextAttributes () const

Returns the text attributes that will be used for displaying the title text.

This is either the text attributes as specified by setTitleTextAttributes, or (if setTitleTextAttributes() was not called) the default text attributes.

See also:
resetTitleTextAttributes, hasDefaultTitleTextAttributes

Definition at line 115 of file KDChartCartesianAxis.cpp.

References d, KDChart::TextAttributes::fontSize(), hasDefaultTitleTextAttributes(), KDChart::TextAttributes::setFontSize(), KDChart::Measure::setValue(), KDChart::AbstractAxis::textAttributes(), and KDChart::Measure::value().

```
116 {
117     if( hasDefaultTitleTextAttributes() ){
118         TextAttributes ta( textAttributes() );
119         Measure me( ta.fontSize() );
120         me.setValue( me.value() * 1.5 );
121         ta.setFontSize( me );
122         return ta;
123     }
124     return d->titleTextAttributes;
125 }
```

7.18.4.61 int AbstractArea::topOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the top edge of the area.

Note:
The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 93 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
94 {
95     // Re-calculate the sizes,
96     // so we also get the amountOf..Overlap members set newly:
97     if( ! doNotRecalculate )
98         sizeHint();
99     return d->amountOfTopOverlap;
100 }
```

7.18.5 Member Data Documentation

7.18.5.1 Q_SIGNALS KDChart::AbstractArea::__pad0__ `[protected, inherited]`

Reimplemented in KDChart::AbstractCoordinatePlane.

Definition at line 141 of file KDChartAbstractArea.h.

7.18.5.2 QWidget∗ KDChart::AbstractLayoutItem::mParent `[protected, inherited]`

Definition at line 88 of file KDChartLayoutItems.h.

Referenced by KDChart::AbstractLayoutItem::setParentWidget().

7.18.5.3 QLayout∗ KDChart::AbstractLayoutItem::mParentLayout `[protected, inherited]`

Definition at line 89 of file KDChartLayoutItems.h.

7.18.5.4 public KDChart::AbstractAxis::Q_SLOTS `[inherited]`

Definition at line 129 of file KDChartAbstractAxis.h.

7.18.5.5 protected KDChart::AbstractAxis::Q_SLOTS `[inherited]`

Definition at line 126 of file KDChartAbstractAxis.h.

The documentation for this class was generated from the following files:

- KDChartCartesianAxis.h
- KDChartCartesianAxis.cpp

7.19 KDChart::CartesianCoordinatePlane Class Reference

`#include <KDChartCartesianCoordinatePlane.h>`

Inheritance diagram for KDChart::CartesianCoordinatePlane:Collaboration diagram for KDChart::CartesianCoordinatePlane:

Public Types

- enum AxesCalcMode {
  Linear,
  Logarithmic }

Public Member Functions

- void addDiagram (AbstractDiagram ∗diagram)

  Adds a diagram to this coordinate plane.

- void adjustHorizontalRangeToData ()

  Adjust horizontal range settings to the ranges covered by the model's data values.

- void adjustVerticalRangeToData ()

  Adjust vertical range settings to the ranges covered by the model's data values.

- void alignToReferencePoint (const RelativePosition &position)
- const bool autoAdjustGridToZoom () const

  Return the status of the built-in grid adjusting feature.

- unsigned int autoAdjustHorizontalRangeToData () const

  Returns the maximal allowed percent of the horizontal space covered by the coordinate plane that may be empty.

- unsigned int autoAdjustVerticalRangeToData () const

  Returns the maximal allowed percent of the vertical space covered by the coordinate plane that may be empty.

- AxesCalcMode axesCalcModeX () const
- AxesCalcMode axesCalcModeY () const
- BackgroundAttributes backgroundAttributes () const
- virtual int bottomOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart:AutoSpacerLayoutItem::sizeHint().

- CartesianCoordinatePlane (Chart ∗parent=0)
- bool compare (const AbstractAreaBase ∗other) const

  Returns true if both areas have the same settings.

- AbstractDiagram ∗ diagram ()
- ConstAbstractDiagramList diagrams () const
- AbstractDiagramList diagrams ()

- bool doesIsometricScaling () const
- virtual Qt::Orientations expandingDirections () const

  pure virtual in QLayoutItem

- FrameAttributes frameAttributes () const
- virtual QRect geometry () const

  pure virtual in QLayoutItem

- void getFrameLeadings (int &left, int &top, int &right, int &bottom) const
- GridAttributes globalGridAttributes () const
- const GridAttributes gridAttributes (Qt::Orientation orientation) const
- DataDimensionsList gridDimensionsList ()

  Returns the dimensions used for drawing the grid lines.

- bool hasOwnGridAttributes (Qt::Orientation orientation) const
- QPair< qreal, qreal > horizontalRange () const
- virtual bool isEmpty () const

  pure virtual in QLayoutItem

- const bool isVisiblePoint (const QPointF &point) const

  Tests, if a point is visible on the coordinate plane.

- void layoutPlanes ()

  Calling layoutPlanes() on the plane triggers the global KDChart::Chart::slotLayoutPlanes().

- virtual int leftOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- virtual QSize maximumSize () const

  pure virtual in QLayoutItem

- virtual QSize minimumSize () const

  pure virtual in QLayoutItem

- virtual QSize minimumSizeHint () const

  [reimplemented]

- void mousePressEvent (QMouseEvent ∗event)

  reimp

- void needLayoutPlanes ()

  Emitted when plane needs to trigger the Chart's layouting of the coord.

- void needRelayout ()

  Emitted when plane needs to trigger the Chart's layouting.

- void needUpdate ()

  Emitted when plane needs to update its drawings.

- virtual void paint (QPainter ∗)

reimpl

- virtual void paintAll (QPainter &painter)

  Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

- virtual void paintBackground (QPainter &painter, const QRect &rectangle)
- virtual void paintCtx (PaintContext ∗context)

  Default impl: Paint the complete item using its layouted position and size.

- virtual void paintFrame (QPainter &painter, const QRect &rectangle)
- virtual void paintIntoRect (QPainter &painter, const QRect &rect)

  Draws the background and frame, then calls paint().

- const Chart ∗ parent () const
- Chart ∗ parent ()
- QLayout ∗ parentLayout ()
- void propertiesChanged ()

  Emitted upon change of a property of the Coordinate Plane or any of its components.

- AbstractCoordinatePlane ∗ referenceCoordinatePlane () const

  There are two ways, in which planes can be caused to interact, in where they are put layouting wise: The first is the reference plane.

- void relayout ()

  Calling relayout() on the plane triggers the global KDChart::Chart::slotRelayout().

- void removeFromParentLayout ()
- virtual void replaceDiagram (AbstractDiagram ∗diagram, AbstractDiagram ∗oldDiagram=0)

  Replaces the old diagram, or appends the diagram, it there is none yet.

- void resetGridAttributes (Qt::Orientation orientation)

  Reset the attributes to be used for grid lines drawn in horizontal direction (or in vertical direction, resp.).

- virtual int rightOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- void setAutoAdjustGridToZoom (bool autoAdjust)

  Disable / re-enable the built-in grid adjusting feature.

- void setAutoAdjustHorizontalRangeToData (unsigned int percentEmpty=67)

  Automatically adjust horizontal range settings to the ranges covered by the model's values, when ever the data have changed, and then emit horizontalRangeAutomaticallyAdjusted.

- void setAutoAdjustVerticalRangeToData (unsigned int percentEmpty=67)

  Automatically adjust vertical range settings to the ranges covered by the model's values, when ever the data have changed, and then emit verticalRangeAutomaticallyAdjusted.

- void setAxesCalcModes (AxesCalcMode mode)

  Specifies the calculation modes for all axes.

- void setAxesCalcModeX (AxesCalcMode mode)

  Specifies the calculation mode for all Abscissa axes.

- void setAxesCalcModeY (AxesCalcMode mode)

  Specifies the calculation mode for all Ordinate axes.

- void setBackgroundAttributes (const BackgroundAttributes &a)
- void setFrameAttributes (const FrameAttributes &a)
- virtual void setGeometry (const QRect &r)

  pure virtual in QLayoutItem

- void setGlobalGridAttributes (const GridAttributes &)

  Set the grid attributes to be used by this coordinate plane.

- void setGridAttributes (Qt::Orientation orientation, const GridAttributes &)

  Set the attributes to be used for grid lines drawn in horizontal direction (or in vertical direction, resp.).

- void setGridNeedsRecalculate ()

  Used by the chart to clear the cached grid data.

- void setHorizontalRange (const QPair< qreal, qreal > &range)

  Set the boundaries of the visible value space displayed in horizontal direction.

- void setIsometricScaling (bool onOff)
- void setParent (Chart ∗parent)

  Called internally by KDChart::Chart.

- void setParentLayout (QLayout ∗lay)
- virtual void setParentWidget (QWidget ∗widget)

  Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

- void setReferenceCoordinatePlane (AbstractCoordinatePlane ∗plane)

  Set another coordinate plane to be used as the reference plane for this one.

- void setVerticalRange (const QPair< qreal, qreal > &range)

  Set the boundaries of the visible value space displayed in vertical direction.

- virtual void setZoomCenter (QPointF center)
- virtual void setZoomFactorX (double factor)
- virtual void setZoomFactorY (double factor)
- virtual QSize sizeHint () const

  pure virtual in QLayoutItem

- virtual void sizeHintChanged () const

  Report changed size hint: ask the parent widget to recalculate the layout.

- virtual QSizePolicy sizePolicy () const

  [reimplemented]

- virtual void takeDiagram (AbstractDiagram ∗diagram)

  Removes the diagram from the plane, without deleting it.

- virtual int topOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- const QPointF translate (const QPointF &diagramPoint) const

  Translate the given point in value space coordinates to a position in pixel space.

- QPair< qreal, qreal > verticalRange () const
- virtual QPointF zoomCenter () const
- virtual double zoomFactorX () const
- virtual double zoomFactorY () const
- ∼CartesianCoordinatePlane ()

Static Public Member Functions

- void paintBackgroundAttributes (QPainter &painter, const QRect &rectangle, const KDChart::BackgroundAttributes &attributes)
- void paintFrameAttributes (QPainter &painter, const QRect &rectangle, const KDChart::FrameAttributes &attributes)

Public Attributes

- Q_SIGNALS __pad0__: void destroyedCoordinatePlane(AbstractCoordinatePlane∗)
- public Q_SLOTS: void adjustRangesToData()

Protected Member Functions

- QRectF adjustedToMaxEmptyInnerPercentage (const QRectF &r, unsigned int percentX, unsigned int percentY) const
- virtual QRect areaGeometry () const
- virtual QRectF calculateRawDataBoundingRect () const
- bool doneSetZoomCenter (QPointF center)
- bool doneSetZoomFactorX (double factor)
- bool doneSetZoomFactorY (double factor)
- virtual QRectF drawingArea () const
- virtual DataDimensionsList getDataDimensionsList () const
- QRectF getRawDataBoundingRectFromDiagrams () const
- QRect innerRect () const
- void layoutDiagrams ()

  Distribute the available space among the diagrams and axes.

- void paintEvent (QPaintEvent ∗)
- virtual void positionHasChanged ()
- const QPointF translateBack (const QPointF &screenPoint) const

Protected Attributes

- QWidget ∗ mParent
- QLayout ∗ mParentLayout
- protected Q_SLOTS: void slotLayoutChanged(AbstractDiagram∗)

7.19.1 Member Enumeration Documentation

7.19.1.1 enum KDChart::AbstractCoordinatePlane::AxesCalcMode `[inherited]`

Enumeration values:

Linear

Logarithmic

Definition at line 55 of file KDChartAbstractCoordinatePlane.h.

```
55 { Linear, Logarithmic };
```

7.19.2 Constructor & Destructor Documentation

7.19.2.1 CartesianCoordinatePlane::CartesianCoordinatePlane (Chart ∗ *parent* = 0) `[explicit]`

Definition at line 67 of file KDChartCartesianCoordinatePlane.cpp.

```
68     : AbstractCoordinatePlane ( new Private(), parent )
69 {
70     // this bloc left empty intentionally
71 }
```

7.19.2.2 CartesianCoordinatePlane::∼CartesianCoordinatePlane ()

Definition at line 73 of file KDChartCartesianCoordinatePlane.cpp.

```
74 {
75     // this bloc left empty intentionally
76 }
```

7.19.3 Member Function Documentation

7.19.3.1 void CartesianCoordinatePlane::addDiagram (AbstractDiagram ∗ *diagram*) `[virtual]`

Adds a diagram to this coordinate plane.

Parameters:

diagram The diagram to add.

See also:

replaceDiagram, takeDiagram

Reimplemented from KDChart::AbstractCoordinatePlane.

Definition at line 84 of file KDChartCartesianCoordinatePlane.cpp.

References KDChart::AbstractCoordinatePlane::addDiagram(), and KDChart::AbstractCoordinatePlane::propertiesChanged().

```
{
    Q_ASSERT_X ( dynamic_cast<AbstractCartesianDiagram*> ( diagram ),
                 "CartesianCoordinatePlane::addDiagram", "Only cartesian "
                 "diagrams can be added to a cartesian coordinate plane!" );
    AbstractCoordinatePlane::addDiagram ( diagram );
    connect ( diagram,  SIGNAL ( layoutChanged ( AbstractDiagram* ) ),
              SLOT ( slotLayoutChanged ( AbstractDiagram* ) ) );

    connect( diagram, SIGNAL( propertiesChanged() ),this, SIGNAL( propertiesChanged() ) );
}
```

7.19.3.2 QRectF CartesianCoordinatePlane::adjustedToMaxEmptyInnerPercentage (const QRectF & *r*, unsigned int *percentX*, unsigned int *percentY*) const `[protected]`

Definition at line 174 of file KDChartCartesianCoordinatePlane.cpp.

Referenced by calculateRawDataBoundingRect().

```
{
    QRectF erg( r );
    if( percentX < 100 || percentX == 1000 ) {
        const bool isPositive = (r.left() >= 0);
        if( (r.right() >= 0) == isPositive ){
            const qreal innerBound =
                    isPositive ? qMin(r.left(), r.right()) : qMax(r.left(), r.right());
            const qreal outerBound =
                    isPositive ? qMax(r.left(), r.right()) : qMin(r.left(), r.right());
            if( innerBound / outerBound * 100 <= percentX )
            {
                if( isPositive )
                    erg.setLeft( 0.0 );
                else
                    erg.setRight( 0.0 );
            }
        }
    }
    if( percentY < 100 || percentY == 1000 ) {
        const bool isPositive = (r.bottom() >= 0);
        if( (r.top() >= 0) == isPositive ){
            const qreal innerBound =
                    isPositive ? qMin(r.top(), r.bottom()) : qMax(r.top(), r.bottom());
            const qreal outerBound =
                    isPositive ? qMax(r.top(), r.bottom()) : qMin(r.top(), r.bottom());
            if( innerBound / outerBound * 100 <= percentY )
            {
                if( isPositive )
                    erg.setBottom( 0.0 );
                else
                    erg.setTop( 0.0 );
            }
        }
    }
    return erg;
}
```

7.19.3.3 void CartesianCoordinatePlane::adjustHorizontalRangeToData ()

Adjust horizontal range settings to the ranges covered by the model's data values.

See also:
adjustRangesToData

Definition at line 583 of file KDChartCartesianCoordinatePlane.cpp.

References d, getRawDataBoundingRectFromDiagrams(), layoutDiagrams(), and KDChart::AbstractCoordinatePlane::propertiesChanged().

```
{
    const QRectF dataBoundingRect( getRawDataBoundingRectFromDiagrams() );
    d->horizontalMin = dataBoundingRect.left();
    d->horizontalMax = dataBoundingRect.right();
    layoutDiagrams();
    emit propertiesChanged();
}
```

7.19.3.4 void CartesianCoordinatePlane::adjustVerticalRangeToData ()

Adjust vertical range settings to the ranges covered by the model's data values.

See also:
adjustRangesToData

Definition at line 592 of file KDChartCartesianCoordinatePlane.cpp.

References d, getRawDataBoundingRectFromDiagrams(), layoutDiagrams(), and KDChart::AbstractCoordinatePlane::propertiesChanged().

```
{
    const QRectF dataBoundingRect( getRawDataBoundingRectFromDiagrams() );
    d->verticalMin = dataBoundingRect.bottom();
    d->verticalMax = dataBoundingRect.top();
    layoutDiagrams();
    emit propertiesChanged();
}
```

7.19.3.5 void AbstractAreaBase::alignToReferencePoint (const RelativePosition & *position*) `[inherited]`

Definition at line 90 of file KDChartAbstractAreaBase.cpp.

```
{
    Q_UNUSED( position );
    // PENDING(kalle) FIXME
    qWarning( "Sorry, not implemented: void AbstractAreaBase::alignToReferencePoint( const RelativePosi
}
```

7.19.3.6 QRect AbstractArea::areaGeometry () const `[protected, virtual, inherited]`

Implements KDChart::AbstractAreaBase.

Definition at line 150 of file KDChartAbstractArea.cpp.

Referenced by drawingArea(), KDChart::PolarCoordinatePlane::layoutDiagrams(), KDChart::CartesianAxis::paint(), KDChart::AbstractArea::paintAll(), and KDChart::CartesianAxis::paintCtx().

```
{
    return geometry();
}
```

7.19.3.7 const bool KDChart::CartesianCoordinatePlane::autoAdjustGridToZoom () const

Return the status of the built-in grid adjusting feature.

See also:
setAutoAdjustGridToZoom

Definition at line 691 of file KDChartCartesianCoordinatePlane.cpp.

References d.

```
{
    return d->autoAdjustGridToZoom;
}
```

7.19.3.8 unsigned int CartesianCoordinatePlane::autoAdjustHorizontalRangeToData () const

Returns the maximal allowed percent of the horizontal space covered by the coordinate plane that may be empty.

Returns:
A percent value indicating how much of the horizontal space may be empty. If more than this is empty, automatic range adjusting is applied. A return value of 100 indicates that no such automatic adjusting is done at all.

See also:
setAutoAdjustHorizontalRangeToData, adjustRangesToData

Definition at line 619 of file KDChartCartesianCoordinatePlane.cpp.

References d.

```
{
    return d->autoAdjustHorizontalRangeToData;
}
```

7.19.3.9 unsigned int CartesianCoordinatePlane::autoAdjustVerticalRangeToData () const

Returns the maximal allowed percent of the vertical space covered by the coordinate plane that may be empty.

Returns:

A percent value indicating how much of the vertical space may be empty. If more than this is empty, automatic range adjusting is applied. A return value of 100 indicates that no such automatic adjusting is done at all.

See also:

setAutoAdjustVerticalRangeToData, adjustRangesToData

Definition at line 624 of file KDChartCartesianCoordinatePlane.cpp.

References d.

```
{
    return d->autoAdjustVerticalRangeToData;
}
```

7.19.3.10 CartesianCoordinatePlane::AxesCalcMode CartesianCoordinatePlane::axesCalcModeX () const

Definition at line 508 of file KDChartCartesianCoordinatePlane.cpp.

References d.

Referenced by getDataDimensionsList().

```
{
    return d->coordinateTransformation.axesCalcModeX;
}
```

7.19.3.11 CartesianCoordinatePlane::AxesCalcMode CartesianCoordinatePlane::axesCalcModeY () const

Definition at line 503 of file KDChartCartesianCoordinatePlane.cpp.

References d.

Referenced by getDataDimensionsList().

```
{
    return d->coordinateTransformation.axesCalcModeY;
}
```

7.19.3.12 BackgroundAttributes AbstractAreaBase::backgroundAttributes () const `[inherited]`

Definition at line 112 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by updateCommonBrush().

```
{
    return d->backgroundAttributes;
}
```

7.19.3.13 int AbstractArea::bottomOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart:AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the bottom edge of the area.

Note:

> The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 101 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
    // Re-calculate the sizes,
    // so we also get the amountOf..Overlap members set newly:
    if( ! doNotRecalculate )
        sizeHint();
    return d->amountOfBottomOverlap;
}
```

7.19.3.14 QRectF CartesianCoordinatePlane::calculateRawDataBoundingRect () const `[protected, virtual]`

Definition at line 214 of file KDChartCartesianCoordinatePlane.cpp.

References adjustedToMaxEmptyInnerPercentage(), d, and getRawDataBoundingRectFromDiagrams().

Referenced by getDataDimensionsList().

```
{
    // are manually set ranges to be applied?
    const bool bAutoAdjustHorizontalRange = (d->autoAdjustHorizontalRangeToData < 100);
    const bool bAutoAdjustVerticalRange   = (d->autoAdjustVerticalRangeToData   < 100);

    const bool bHardHorizontalRange = (d->horizontalMin != d->horizontalMax) && ! bAutoAdjustHorizonta
    const bool bHardVerticalRange   = (d->verticalMin   != d->verticalMax)   && ! bAutoAdjustVerticalR
    QRectF dataBoundingRect;

    // if custom boundaries are set on the plane, use them
    if ( bHardHorizontalRange && bHardVerticalRange ) {
        dataBoundingRect.setLeft(   d->horizontalMin );
        dataBoundingRect.setRight(  d->horizontalMax );
        dataBoundingRect.setBottom( d->verticalMin );
        dataBoundingRect.setTop(    d->verticalMax );
    }else{
        // determine unit of the rectangles of all involved diagrams:
        dataBoundingRect = getRawDataBoundingRectFromDiagrams();
        if ( bHardHorizontalRange ) {
            dataBoundingRect.setLeft(  d->horizontalMin );
            dataBoundingRect.setRight( d->horizontalMax );
        }
        if ( bHardVerticalRange ) {
            dataBoundingRect.setBottom( d->verticalMin );
            dataBoundingRect.setTop(    d->verticalMax );
        }
    }
    // recalculate the bounds, if automatic adjusting of ranges is desired AND
    //                         both bounds are at the same side of the zero line
    dataBoundingRect = adjustedToMaxEmptyInnerPercentage(
            dataBoundingRect, d->autoAdjustHorizontalRangeToData, d->autoAdjustVerticalRangeToData );
    if( bAutoAdjustHorizontalRange ){
        const_cast<CartesianCoordinatePlane::Private *>(d)->horizontalMin = dataBoundingRect.left();
        const_cast<CartesianCoordinatePlane::Private *>(d)->horizontalMax = dataBoundingRect.right();
    }
    if( bAutoAdjustVerticalRange ){
        const_cast<CartesianCoordinatePlane*>(this)->d->verticalMin = dataBoundingRect.bottom();
        const_cast<CartesianCoordinatePlane*>(this)->d->verticalMax = dataBoundingRect.top();
    }
    //qDebug() << "CartesianCoordinatePlane::calculateRawDataBoundingRect()\nreturns data boundaries:
    return dataBoundingRect;
}
```

7.19.3.15 bool AbstractAreaBase::compare (const AbstractAreaBase ∗ *other*) const `[inherited]`

Returns true if both areas have the same settings.

Definition at line 75 of file KDChartAbstractAreaBase.cpp.

```
{
    if( other == this ) return true;
    if( ! other ){
        //qDebug() << "CartesianAxis::compare() cannot compare to Null pointer";
        return false;
    }
    /*
    qDebug() << "AbstractAreaBase:" << (frameAttributes() == other->frameAttributes())
        << (backgroundAttributes() == other->backgroundAttributes()) << "\n";
    */
    return  (frameAttributes()      == other->frameAttributes()) &&
            (backgroundAttributes() == other->backgroundAttributes());
}
```

7.19.3.16 AbstractDiagram ∗ AbstractCoordinatePlane::diagram () `[inherited]`

Returns:

The first diagram associated with this coordinate plane.

Definition at line 113 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::Widget::diagram(), KDChart::Chart::mousePressEvent(), and KDChart::PolarCoordinatePlane::setStartPosition().

```
{
    if ( d->diagrams.isEmpty() )
116     {
117         return 0;
118     } else {
119         return d->diagrams.first();
120     }
121 }
```

7.19.3.17 ConstAbstractDiagramList AbstractCoordinatePlane::diagrams () const `[inherited]`

Returns:

The list of diagrams associated with this coordinate plane.

Definition at line 128 of file KDChartAbstractCoordinatePlane.cpp.

References KDChart::ConstAbstractDiagramList, and d.

```
129 {
130     ConstAbstractDiagramList list;
131 #ifndef QT_NO_STL
132     qCopy( d->diagrams.begin(), d->diagrams.end(), std::back_inserter( list ) );
133 #else
134     Q_FOREACH( AbstractDiagram * a, d->diagrams )
135         list.push_back( a );
136 #endif
137     return list;
138 }
```

7.19.3.18 AbstractDiagramList AbstractCoordinatePlane::diagrams () `[inherited]`

Returns:

The list of diagrams associated with this coordinate plane.

Definition at line 123 of file KDChartAbstractCoordinatePlane.cpp.

References KDChart::AbstractDiagramList, and d.

Referenced by getDataDimensionsList(), getRawDataBoundingRectFromDiagrams(), KDChart::PolarCoordinatePlane::layoutDiagrams(), layoutDiagrams(), KDChart::Chart::mousePressEvent(), KDChart::PolarCoordinatePlane::paint(), and paint().

```
124 {
125     return d->diagrams;
126 }
```

7.19.3.19 bool CartesianCoordinatePlane::doesIsometricScaling () const

Definition at line 428 of file KDChartCartesianCoordinatePlane.cpp.

References d.

```
429 {
430     return d->isometricScaling;
431 }
```

7.19.3.20 bool CartesianCoordinatePlane::doneSetZoomCenter (QPointF *center*) `[protected]`

Definition at line 455 of file KDChartCartesianCoordinatePlane.cpp.

References d.

Referenced by setZoomCenter().

```
{
    bool bDone = ( d->coordinateTransformation.zoom.center() != point );
    if( bDone ){
        d->coordinateTransformation.zoom.setCenter( point );
        if( d->autoAdjustGridToZoom )
            d->grid->setNeedRecalculate();
    }
    return bDone;
}
```

7.19.3.21 bool CartesianCoordinatePlane::doneSetZoomFactorX (double *factor*) `[protected]`

Definition at line 433 of file KDChartCartesianCoordinatePlane.cpp.

References d.

Referenced by setZoomFactorX().

```
{
    bool bDone = ( d->coordinateTransformation.zoom.xFactor != factor );
    if( bDone ){
        d->coordinateTransformation.zoom.xFactor = factor;
        if( d->autoAdjustGridToZoom )
            d->grid->setNeedRecalculate();
    }
    return bDone;
}
```

7.19.3.22 bool CartesianCoordinatePlane::doneSetZoomFactorY (double *factor*) `[protected]`

Definition at line 444 of file KDChartCartesianCoordinatePlane.cpp.

References d.

Referenced by setZoomFactorY().

```
{
    bool bDone = ( d->coordinateTransformation.zoom.yFactor != factor );
    if( bDone ){
        d->coordinateTransformation.zoom.yFactor = factor;
        if( d->autoAdjustGridToZoom )
            d->grid->setNeedRecalculate();
    }
    return bDone;
}
```

7.19.3.23 QRectF CartesianCoordinatePlane::drawingArea () const `[protected, virtual]`

Definition at line 309 of file KDChartCartesianCoordinatePlane.cpp.

References KDChart::AbstractArea::areaGeometry().

Referenced by layoutDiagrams(), and paint().

```
{
    const QRect rect( areaGeometry() );
    return QRectF ( rect.left()+1, rect.top()+1, rect.width() - 3, rect.height() - 3 );
}
```

7.19.3.24 Qt::Orientations KDChart::AbstractCoordinatePlane::expandingDirections () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 208 of file KDChartAbstractCoordinatePlane.cpp.

```
{
    return Qt::Vertical | Qt::Horizontal;
}
```

7.19.3.25 FrameAttributes AbstractAreaBase::frameAttributes () const `[inherited]`

Definition at line 102 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone(), and updateCommonBrush().

```
{
    return d->frameAttributes;
}
```

7.19.3.26 QRect KDChart::AbstractCoordinatePlane::geometry () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 242 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::Chart::mousePressEvent(), and KDChart::PolarCoordinatePlane::paint().

```
{
    return d->geometry;
}
```

7.19.3.27 DataDimensionsList CartesianCoordinatePlane::getDataDimensionsList () const `[protected, virtual]`

Implements KDChart::AbstractCoordinatePlane.

Definition at line 259 of file KDChartCartesianCoordinatePlane.cpp.

References axesCalcModeX(), axesCalcModeY(), calculateRawDataBoundingRect(), KDChart::DataDimensionsList, KDChart::AbstractDiagram::datasetDimension(), KDChart::AbstractCoordinatePlane::diagrams(), gridAttributes(), KDChart::GridAttributes::gridGranularitySequence(), KDChart::GridAttributes::gridStepWidth(), KDChart::GridAttributes::gridSubStepWidth(), and KDChart::AbstractDiagram::percentMode().

```
{

    DataDimensionsList l;
    const AbstractCartesianDiagram* dgr
        = diagrams().isEmpty() ? 0 : dynamic_cast<const AbstractCartesianDiagram*> (diagrams().first()

    if( dgr ){
        const QRectF r( calculateRawDataBoundingRect() );
        // note:
        // We do *not* access d->gridAttributesHorizontal here, but
        // we use the getter function, to get the global attrs, if no
        // special ones have been set for the respective orientation.
        const GridAttributes gaH( gridAttributes( Qt::Horizontal ) );
        const GridAttributes gaV( gridAttributes( Qt::Vertical ) );
        // append the first dimension: for Abscissa axes
        l.append(
            DataDimension(
                r.left(), r.right(),
                dgr->datasetDimension() > 1,
                axesCalcModeX(),
                gaH.gridGranularitySequence(),
                gaH.gridStepWidth(),
                gaH.gridSubStepWidth() ) );
        // append the second dimension: for Ordinate axes
        if( dgr->percentMode() )
            l.append(
                DataDimension(
                    // always return 0-100 when in percentMode
                    0.0, 100.0,
                    true,
                    axesCalcModeY(),
                    KDChartEnums::GranularitySequence_10_20,
                    10.0 ) );
        else
            l.append(
                DataDimension(
                    r.bottom(), r.top(),
                    true,
                    axesCalcModeY(),
                    gaV.gridGranularitySequence(),
                    gaV.gridStepWidth(),
                    gaV.gridSubStepWidth() ) );
    }else{
        l.append( DataDimension() ); // This gets us the default 1..0 / 1..0 grid
        l.append( DataDimension() ); // shown, if there is no diagram on this plane.
    }
    return l;
}
```

7.19.3.28 void AbstractAreaBase::getFrameLeadings (int & *left*, int & *top*, int & *right*, int & *bottom*) const `[inherited]`

Definition at line 204 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::AbstractAreaBase::innerRect(), and KDChart::AbstractAreaWidget::paintAll().

```
205 {
206     if( d && d->frameAttributes.isVisible() ){
207         const int padding = qMax( d->frameAttributes.padding(), 0 );
208         left   = padding;
209         top    = padding;
210         right  = padding;
211         bottom = padding;
212     }else{
213         left   = 0;
214         top    = 0;
215         right  = 0;
216         bottom = 0;
217     }
218 }
```

7.19.3.29 QRectF CartesianCoordinatePlane::getRawDataBoundingRectFromDiagrams () const `[protected]`

Definition at line 151 of file KDChartCartesianCoordinatePlane.cpp.

References KDChart::AbstractDiagram::dataBoundaries(), and KDChart::AbstractCoordinatePlane::diagrams().

Referenced by adjustHorizontalRangeToData(), adjustVerticalRangeToData(), and calculateRawDataBoundingRect().

```
152 {
153     // determine unit of the rectangles of all involved diagrams:
154     qreal minX, maxX, minY, maxY;
155     bool bStarting = true;
156     Q_FOREACH( const AbstractDiagram* diagram, diagrams() )
157     {
158         QPair<QPointF, QPointF> dataBoundariesPair = diagram->dataBoundaries();
159         //qDebug() << "CartesianCoordinatePlane::getRawDataBoundingRectFromDiagrams()\ngets diagram->d
160         if ( bStarting || dataBoundariesPair.first.x()  < minX ) minX = dataBoundariesPair.first.x();
161         if ( bStarting || dataBoundariesPair.first.y()  < minY ) minY = dataBoundariesPair.first.y();
162         if ( bStarting || dataBoundariesPair.second.x() > maxX ) maxX = dataBoundariesPair.second.x();
163         if ( bStarting || dataBoundariesPair.second.y() > maxY ) maxY = dataBoundariesPair.second.y();
164         bStarting = false;
165     }
166     //qDebug() << "CartesianCoordinatePlane::getRawDataBoundingRectFromDiagrams()\nreturns data bounda
167     QRectF dataBoundingRect;
168     dataBoundingRect.setBottomLeft( QPointF(minX, minY) );
169     dataBoundingRect.setTopRight(   QPointF(maxX, maxY) );
170     return dataBoundingRect;
171 }
```

7.19.3.30 GridAttributes KDChart::AbstractCoordinatePlane::globalGridAttributes () const `[inherited]`

Returns:

The grid attributes used by this coordinate plane.

See also:

setGlobalGridAttributes
CartesianCoordinatePlane::gridAttributes

Definition at line 157 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::PolarCoordinatePlane::gridAttributes(), and gridAttributes().

```
{
    return d->gridAttributes;
}
```

7.19.3.31 const GridAttributes KDChart::CartesianCoordinatePlane::gridAttributes (Qt::Orientation *orientation*) const

Returns:

The attributes used for grid lines drawn in horizontal direction (or in vertical direction, resp.).

Note:

This function always returns a valid set of grid attributes: If no special grid attributes were set foe this orientation the global attributes are returned, as returned by AbstractCoordinatePlane::globalGridAttributes.

See also:

setGridAttributes
resetGridAttributes
AbstractCoordinatePlane::globalGridAttributes
hasOwnGridAttributes

Definition at line 650 of file KDChartCartesianCoordinatePlane.cpp.

References d, KDChart::AbstractCoordinatePlane::globalGridAttributes(), and hasOwnGridAttributes().

Referenced by getDataDimensionsList().

```
{
    if( hasOwnGridAttributes( orientation ) ){
        if( orientation == Qt::Horizontal )
            return d->gridAttributesHorizontal;
        else
            return d->gridAttributesVertical;
    }else{
        return globalGridAttributes();
    }
}
```

7.19.3.32 KDChart::DataDimensionsList KDChart::AbstractCoordinatePlane::gridDimensionsList () `[inherited]`

Returns the dimensions used for drawing the grid lines.

Returned data is the result of (cached) grid calculations, so - if you need that information for your own tasks - make sure to call again this function after every data modification that has changed the data range, since grid calculation is based upon the data range, thus the grid start/end might have changed if the data was changed.

Note:

Returned list will contain different numbers of DataDimension, depending on the kind of coordinate plane used. For CartesianCoordinatePlane two DataDimension are returned: the first representing grid lines in X direction (matching the Abscissa axes) and the second indicating vertical grid lines (or Ordinate axes, resp.).

Returns:

The dimensions used for drawing the grid lines.

See also:

DataDimension

Definition at line 162 of file KDChartAbstractCoordinatePlane.cpp.

References d, and KDChart::DataDimensionsList.

Referenced by layoutDiagrams(), KDChart::CartesianAxis::maximumSize(), and KDChart::CartesianAxis::paintCtx().

```
163 {
164     //KDChart::DataDimensionsList l( d->grid->updateData( this ) );
165     //qDebug() << "AbstractCoordinatePlane::gridDimensionsList() Y-range:" << l.last().end - l.last().
166     //qDebug() << "AbstractCoordinatePlane::gridDimensionsList() X-range:" << l.first().end - l.first(
167     return d->grid->updateData( this );
168 }
```

7.19.3.33 bool KDChart::CartesianCoordinatePlane::hasOwnGridAttributes (Qt::Orientation *orientation*) const

Returns:

Returns whether the grid attributes have been set for the respective direction via setGridAttributes(orientation).

If false, the grid will use the global attributes set by AbstractCoordinatePlane::globalGridAttributes (or the default attributes, resp.)

See also:

setGridAttributes
resetGridAttributes
AbstractCoordinatePlane::globalGridAttributes

Definition at line 673 of file KDChartCartesianCoordinatePlane.cpp.

References d.

Referenced by gridAttributes().

```
675 {
676     return
677         ( orientation == Qt::Horizontal )
678         ? d->hasOwnGridAttributesHorizontal
679         : d->hasOwnGridAttributesVertical;
680 }
```

7.19.3.34 QPair< qreal, qreal > KDChart::CartesianCoordinatePlane::horizontalRange () const

Returns:

The largest and smallest visible horizontal value space value. If this is not explicitely set,or if both values are the same, the plane will use the union of the dataBoundaries of all associated diagrams.

See also:

KDChart::AbstractDiagram::dataBoundaries

Definition at line 562 of file KDChartCartesianCoordinatePlane.cpp.

References d.

```
{
    return QPair<qreal, qreal>( d->horizontalMin, d->horizontalMax );
}
```

7.19.3.35 QRect AbstractAreaBase::innerRect () const `[protected, inherited]`

Definition at line 220 of file KDChartAbstractAreaBase.cpp.

References KDChart::AbstractAreaBase::areaGeometry(), and KDChart::AbstractAreaBase::getFrameLeadings().

Referenced by KDChart::TextArea::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    int left;
    int top;
    int right;
    int bottom;
    getFrameLeadings( left, top, right, bottom );
    return
        QRect( QPoint(0,0), areaGeometry().size() )
            .adjusted( left, top, -right, -bottom );
}
```

7.19.3.36 bool KDChart::AbstractCoordinatePlane::isEmpty () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 201 of file KDChartAbstractCoordinatePlane.cpp.

```
{
    return false; // never empty!
    // coordinate planes with no associated diagrams
    // are showing a default grid of ()1..10, 1..10) stepWidth 1
}
```

7.19.3.37 const bool KDChart::AbstractCoordinatePlane::isVisiblePoint (const QPointF & *point*) const `[inherited]`

Tests, if a point is visible on the coordinate plane.

Note:

Before calling this function the point must have been translated into coordinate plane space.

Definition at line 275 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
{
    return d->isVisiblePoint( this, point );
}
```

7.19.3.38 void CartesianCoordinatePlane::layoutDiagrams () `[protected, virtual]`

Distribute the available space among the diagrams and axes.

Implements KDChart::AbstractCoordinatePlane.

Definition at line 316 of file KDChartCartesianCoordinatePlane.cpp.

References d, KDChart::DataDimensionsList, KDChart::AbstractCoordinatePlane::diagrams(), KDChart::DataDimension::distance(), drawingArea(), KDChart::DataDimension::end, KDChart::AbstractCoordinatePlane::gridDimensionsList(), and KDChart::DataDimension::start.

Referenced by adjustHorizontalRangeToData(), adjustVerticalRangeToData(), setAutoAdjustHorizontalRangeToData(), setAutoAdjustVerticalRangeToData(), setHorizontalRange(), setIsometricScaling(), and setVerticalRange().

```
{
    //qDebug("KDChart::CartesianCoordinatePlane::layoutDiagrams() called");
    if ( diagrams().isEmpty() )
    {   // FIXME evaluate what can still be prepared
        // FIXME decide default dimension if no diagrams are present (to make empty planes useable)
    }
    // the rectangle the diagrams cover in the *plane*:
    // (Why -3? We save 1px on each side for the antialiased drawing, and
    // respect the way QPainter calculates the width of a painted rect (the
    // size is the rectangle size plus the pen width). This way, most clipping
    // for regular pens should be avoided. When pens with a penWidth or larger
    // than 1 are used, this may not be sufficient.
    const QRectF drawArea( drawingArea() );
    //qDebug() << "drawingArea() returns" << drawArea;

    const DataDimensionsList dimensions( gridDimensionsList() );
    // test for programming errors: critical
    Q_ASSERT_X ( dimensions.count() == 2, "CartesianCoordinatePlane::layoutDiagrams",
                 "Error: gridDimensionsList() did not return exactly two dimensions." );
    const DataDimension dimX = dimensions.first();
    const DataDimension dimY = dimensions.last();
    const qreal distX = dimX.distance();
    const qreal distY = dimY.distance();
    //qDebug() << distX << distY;
    const QPointF pt(qMin(dimX.start, dimX.end), qMax(dimY.start, dimY.end));
    const QSizeF siz( qAbs(distX), -qAbs(distY) );
    const QRectF dataBoundingRect( pt, siz );
    //qDebug() << "dataBoundingRect" << dataBoundingRect;

    // calculate the remaining rectangle, and use it as the diagram area:
    QRectF diagramArea = drawArea;
    diagramArea.setTopLeft ( QPointF ( drawArea.left(), drawArea.top() ) );
    diagramArea.setBottomRight ( QPointF ( drawArea.right(), drawArea.bottom() ) );

    // determine coordinate transformation:
    QPointF diagramTopLeft = dataBoundingRect.topLeft();
    double diagramWidth = dataBoundingRect.width();
    double diagramHeight = dataBoundingRect.height();
    double planeWidth = diagramArea.width();
    double planeHeight = diagramArea.height();
    double scaleX;
    double scaleY;

    double diagramXUnitInCoordinatePlane;
    double diagramYUnitInCoordinatePlane;

    diagramXUnitInCoordinatePlane = diagramWidth != 0 ? planeWidth / diagramWidth : 1;
    diagramYUnitInCoordinatePlane = diagramHeight != 0 ? planeHeight / diagramHeight : 1;
    // calculate isometric scaling factor to maxscale the diagram into
    // the coordinate system:
    if ( d->isometricScaling )
    {
        double scale = qMin ( qAbs ( diagramXUnitInCoordinatePlane ),
                              qAbs ( diagramYUnitInCoordinatePlane ) );

        scaleX = qAbs( scale / diagramXUnitInCoordinatePlane );
        scaleY = qAbs( scale / diagramYUnitInCoordinatePlane );
    } else {
        scaleX = 1.0;
        scaleY = 1.0;
    }

    // calculate diagram origin in plane coordinates:
    QPointF coordinateOrigin = QPointF (
            diagramTopLeft.x() * -diagramXUnitInCoordinatePlane,
    diagramTopLeft.y() * -diagramYUnitInCoordinatePlane );
    coordinateOrigin += diagramArea.topLeft();

    d->coordinateTransformation.originTranslation = coordinateOrigin;

    d->coordinateTransformation.diagramRect = dataBoundingRect;

    d->coordinateTransformation.unitVectorX = diagramXUnitInCoordinatePlane;
    d->coordinateTransformation.unitVectorY = diagramYUnitInCoordinatePlane;

    d->coordinateTransformation.isoScaleX = scaleX;
    d->coordinateTransformation.isoScaleY = scaleY;

    //      adapt diagram area to effect of isometric scaling:
    diagramArea.setTopLeft( translate ( dataBoundingRect.topLeft() ) );
    diagramArea.setBottomRight ( translate ( dataBoundingRect.bottomRight() ) );

    //qDebug("KDChart::CartesianCoordinatePlane::layoutDiagrams() done,\ncalling update() now:");
    update();
}
```

7.19.3.39 void KDChart::AbstractCoordinatePlane::layoutPlanes () `[inherited]`

Calling layoutPlanes() on the plane triggers the global KDChart::Chart::slotLayoutPlanes().

Definition at line 259 of file KDChartAbstractCoordinatePlane.cpp.

References KDChart::AbstractCoordinatePlane::needLayoutPlanes().

Referenced by KDChart::AbstractCoordinatePlane::addDiagram(), KDChart::CartesianAxis::layoutPlanes(), KDChart::AbstractCartesianDiagram::layoutPlanes(), and KDChart::AbstractCoordinatePlane::replaceDiagram().

```
{
    //qDebug("KDChart::AbstractCoordinatePlane::relayout() called");
    emit needLayoutPlanes();
}
```

7.19.3.40 int AbstractArea::leftOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the left edge of the area.

Note:

The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 77 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
    // Re-calculate the sizes,
    // so we also get the amountOf..Overlap members set newly:
    if( ! doNotRecalculate )
        sizeHint();
    return d->amountOfLeftOverlap;
}
```

7.19.3.41 QSize KDChart::AbstractCoordinatePlane::maximumSize () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 213 of file KDChartAbstractCoordinatePlane.cpp.

Referenced by KDChart::AbstractCoordinatePlane::sizeHint().

```
{
    // No maximum size set. Especially not parent()->size(), we are not layouting
    // to the parent widget's size when using Chart::paint()!
    return QSize(QLAYOUTSIZE_MAX, QLAYOUTSIZE_MAX);
}
```

7.19.3.42 QSize KDChart::AbstractCoordinatePlane::minimumSize () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 220 of file KDChartAbstractCoordinatePlane.cpp.

```
{
    return QSize(60, 60); // this default can be overwritten by derived classes
}
```

7.19.3.43 QSize KDChart::AbstractCoordinatePlane::minimumSizeHint () const `[virtual, inherited]`

[reimplemented]

Definition at line 140 of file KDChartAbstractCoordinatePlane.cpp.

```
{
    return QSize( 200, 200 );
}
```

7.19.3.44 void KDChart::AbstractCoordinatePlane::mousePressEvent (QMouseEvent ∗ *event*) `[inherited]`

reimp

Definition at line 266 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::Chart::mousePressEvent().

```
{
    KDAB_FOREACH( AbstractDiagram * a, d->diagrams )
    {
        a->mousePressEvent( event );
    }
}
```

7.19.3.45 void KDChart::AbstractCoordinatePlane::needLayoutPlanes () `[inherited]`

Emitted when plane needs to trigger the Chart's layouting of the coord.

planes.

Referenced by KDChart::AbstractCoordinatePlane::layoutPlanes().

7.19.3.46 void KDChart::AbstractCoordinatePlane::needRelayout () `[inherited]`

Emitted when plane needs to trigger the Chart's layouting.

Referenced by KDChart::AbstractCoordinatePlane::relayout().

7.19.3.47 void KDChart::AbstractCoordinatePlane::needUpdate () `[inherited]`

Emitted when plane needs to update its drawings.

7.19.3.48 void CartesianCoordinatePlane::paint (QPainter ∗) `[virtual]`

reimpl

Implements KDChart::AbstractLayoutItem.

Definition at line 97 of file KDChartCartesianCoordinatePlane.cpp.

References KDChart::AbstractDiagramList, d, KDChart::AbstractCoordinatePlane::diagrams(), drawingArea(), KDChart::PaintContext::setCoordinatePlane(), KDChart::PaintContext::setPainter(), and KDChart::PaintContext::setRectangle().

```
{
    // prevent recursive call:
    //qDebug("attempt plane::paint()");
    if( d->bPaintIsRunning ){
        return;
    }
    d->bPaintIsRunning = true;

    //qDebug() << "start plane::paint()";

    AbstractDiagramList diags = diagrams();
    if ( !diags.isEmpty() )
    {
        PaintContext ctx;
        ctx.setPainter ( painter );
        ctx.setCoordinatePlane ( this );
        const QRectF drawArea( drawingArea() );
        ctx.setRectangle ( drawArea );

        // enabling clipping so that we’re not drawing outside
        PainterSaver painterSaver( painter );
        QRect clipRect = drawArea.toRect().adjusted( -1, -1, 1, 1 );
        QRegion clipRegion( clipRect );
        painter->setClipRegion( clipRegion );

        // paint the coordinate system rulers:
        d->grid->drawGrid( &ctx );

        // paint the diagrams:
        for ( int i = 0; i < diags.size(); i++ )
        {
//qDebug("  start diags[i]->paint ( &ctx );");
            PainterSaver diagramPainterSaver( painter );
            diags[i]->paint ( &ctx );
//qDebug("  done: diags[i]->paint ( &ctx );");
        }

        //for debugging:
        //    painter->drawRect( drawArea.adjusted(4,4,-4,-4) );
        //    painter->drawRect( drawArea.adjusted(2,2,-2,-2) );
        //    painter->drawRect( drawArea );
    }
    d->bPaintIsRunning = false;
    //qDebug("done: plane::paint()");
}
```

7.19.3.49 void AbstractArea::paintAll (QPainter & *painter*) [virtual, inherited]

Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

Reimplemented from KDChart::AbstractLayoutItem.

Definition at line 123 of file KDChartAbstractArea.cpp.

References KDChart::AbstractArea::areaGeometry(), d, KDChart::AbstractAreaBase::innerRect(), KDChart::AbstractLayoutItem::paint(), KDChart::AbstractAreaBase::paintBackground(), and KDChart::AbstractAreaBase::paintFrame().

Referenced by KDChart::AbstractArea::paintIntoRect().

```
{
    // Paint the background and frame
    const QRect overlappingArea( geometry().adjusted(
            -d->amountOfLeftOverlap,
            -d->amountOfTopOverlap,
            d->amountOfRightOverlap,
            d->amountOfBottomOverlap ) );
    paintBackground( painter, overlappingArea );
    paintFrame(      painter, overlappingArea );

    // temporarily adjust the widget size, to be sure all content gets calculated
    // to fit into the inner rectangle
    const QRect oldGeometry( areaGeometry()  );
    QRect inner( innerRect() );
    inner.moveTo(
        oldGeometry.left() + inner.left(),
        oldGeometry.top()  + inner.top() );
    const bool needAdjustGeometry = oldGeometry != inner;
    if( needAdjustGeometry )
        setGeometry( inner );
    paint( &painter );
    if( needAdjustGeometry )
        setGeometry( oldGeometry );
    //qDebug() << "AbstractAreaWidget::paintAll() done.";
}
```

7.19.3.50 void AbstractAreaBase::paintBackground (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 188 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintBackgroundAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintBackground()",
                "Private class was not initialized!" );
    paintBackgroundAttributes( painter, rect, d->backgroundAttributes );
}
```

7.19.3.51 void AbstractAreaBase::paintBackgroundAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::BackgroundAttributes & *attributes*) `[static, inherited]`

Definition at line 119 of file KDChartAbstractAreaBase.cpp.

References KDChart::BackgroundAttributes::brush(), KDChart::BackgroundAttributes::isVisible(), KDChart::BackgroundAttributes::pixmap(), and KDChart::BackgroundAttributes::pixmapMode().

Referenced by KDChart::AbstractAreaBase::paintBackground().

```
{
    if( !attributes.isVisible() ) return;

    /* first draw the brush (may contain a pixmap)*/
    if( Qt::NoBrush != attributes.brush().style() ) {
        KDChart::PainterSaver painterSaver( &painter );
        painter.setPen( Qt::NoPen );
        const QPointF newTopLeft( painter.deviceMatrix().map( rect.topLeft() ) );
        painter.setBrushOrigin( newTopLeft );
        painter.setBrush( attributes.brush() );
        painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
    }
    /* next draw the backPixmap over the brush */
    if( !attributes.pixmap().isNull() &&
        attributes.pixmapMode() != BackgroundAttributes::BackgroundPixmapModeNone ) {
        QPointF ol = rect.topLeft();
        if( BackgroundAttributes::BackgroundPixmapModeCentered == attributes.pixmapMode() )
        {
            ol.setX( rect.center().x() - attributes.pixmap().width() / 2 );
            ol.setY( rect.center().y() - attributes.pixmap().height()/ 2 );
            painter.drawPixmap( ol, attributes.pixmap() );
        } else {
            QMatrix m;
            double zW = (double)rect.width()  / (double)attributes.pixmap().width();
            double zH = (double)rect.height() / (double)attributes.pixmap().height();
            switch( attributes.pixmapMode() ) {
            case BackgroundAttributes::BackgroundPixmapModeScaled:
            {
                double z;
                z = qMin( zW, zH );
                m.scale( z, z );
            }
            break;
            case BackgroundAttributes::BackgroundPixmapModeStretched:
                m.scale( zW, zH );
                break;
            default:
                ; // Cannot happen, previously checked
            }
            QPixmap pm = attributes.pixmap().transformed( m );
            ol.setX( rect.center().x() - pm.width() / 2 );
            ol.setY( rect.center().y() - pm.height()/ 2 );
            painter.drawPixmap( ol, pm );
        }
    }
}
```

7.19.3.52 void KDChart::AbstractLayoutItem::paintCtx (PaintContext ∗ *context*) `[virtual, inherited]`

Default impl: Paint the complete item using its layouted position and size.

Reimplemented in KDChart::CartesianAxis.

Definition at line 77 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint(), and KDChart::PaintContext::painter().

```
78 {
79     if( context )
80         paint( context->painter() );
81 }
```

7.19.3.53 void KDChart::CartesianCoordinatePlane::paintEvent (QPaintEvent ∗) `[protected]`

7.19.3.54 void AbstractAreaBase::paintFrame (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 196 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintFrameAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintFrame()",
                 "Private class was not initialized!" );
    paintFrameAttributes( painter, rect, d->frameAttributes );
}
```

7.19.3.55 void AbstractAreaBase::paintFrameAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::FrameAttributes & *attributes*) `[static, inherited]`

Definition at line 169 of file KDChartAbstractAreaBase.cpp.

References KDChart::FrameAttributes::isVisible(), and KDChart::FrameAttributes::pen().

Referenced by KDChart::AbstractAreaBase::paintFrame().

```
{

    if( !attributes.isVisible() ) return;

    // Note: We set the brush to NoBrush explicitely here.
    //       Otherwise we might get a filled rectangle, so any
    //       previously drawn background would be overwritten by that area.

    const QPen   oldPen(   painter.pen() );
    const QBrush oldBrush( painter.brush() );
    painter.setPen(   attributes.pen() );
    painter.setBrush( Qt::NoBrush );
    painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
    painter.setBrush( oldBrush );
    painter.setPen(   oldPen );
}
```

7.19.3.56 void AbstractArea::paintIntoRect (QPainter & *painter*, const QRect & *rect*) `[virtual, inherited]`

Draws the background and frame, then calls paint().

In most cases there is no need to overwrite this method in a derived class, but you would overwrite AbstractLayoutItem::paint() instead.

Definition at line 111 of file KDChartAbstractArea.cpp.

References KDChart::AbstractArea::paintAll().

```
{
    const QRect oldGeometry( geometry() );
    if( oldGeometry != rect )
        setGeometry( rect );
    painter.translate( rect.left(), rect.top() );
    paintAll( painter );
    painter.translate( -rect.left(), -rect.top() );
    if( oldGeometry != rect )
        setGeometry( oldGeometry );
}
```

7.19.3.57 const KDChart::Chart ∗ KDChart::AbstractCoordinatePlane::parent () const `[inherited]`

Definition at line 190 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
{
    return d->parent;
}
```

7.19.3.58 KDChart::Chart ∗ KDChart::AbstractCoordinatePlane::parent () `[inherited]`

Definition at line 195 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::CartesianAxis::maximumSize(), and KDChart::CartesianAxis::paintCtx().

```
{
    return d->parent;
}
```

7.19.3.59 QLayout∗ KDChart::AbstractLayoutItem::parentLayout () `[inherited]`

Definition at line 74 of file KDChartLayoutItems.h.

```
        {
            return mParentLayout;
        }
```

7.19.3.60 void AbstractArea::positionHasChanged () `[protected, virtual, inherited]`

Reimplemented from KDChart::AbstractAreaBase.

Definition at line 155 of file KDChartAbstractArea.cpp.

```
{
    emit positionChanged( this );
}
```

7.19.3.61 void KDChart::AbstractCoordinatePlane::propertiesChanged () `[inherited]`

Emitted upon change of a property of the Coordinate Plane or any of its components.

Referenced by addDiagram(), adjustHorizontalRangeToData(), adjustVerticalRangeToData(), setAutoAdjustGridToZoom(), setAutoAdjustHorizontalRangeToData(), setAutoAdjustVerticalRangeToData(), setAxesCalcModes(), setAxesCalcModeX(), setAxesCalcModeY(), KDChart::PolarCoordinatePlane::setGridAttributes(), setGridAttributes(), setHorizontalRange(), setIsometricScaling(), setVerticalRange(), setZoomCenter(), setZoomFactorX(), and setZoomFactorY().

7.19.3.62 AbstractCoordinatePlane ∗ KDChart::AbstractCoordinatePlane::referenceCoordinatePlane () const `[inherited]`

There are two ways, in which planes can be caused to interact, in where they are put layouting wise: The first is the reference plane.

If such a reference plane is set, on a plane, it will use the same cell in the layout as that one. In addition to this, planes can share an axis. In that case they will be layed out in relation to each other as suggested by the position of the axis. If, for example Plane1 and Plane2 share an axis at position Left, that will result in the layout: Axis Plane1 Plane 2, vertically. If Plane1 also happens to be Plane2's reference plane, both planes are drawn over each other. The reference plane concept allows two planes to share the same space even if neither has any axis, and in case there are shared axis, it is used to decided, whether the planes should be painted on top of each other or layed out vertically or horizontally next to each other.

Returns:

The reference coordinate plane associated with this one.

Definition at line 180 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
{
    return d->referenceCoordinatePlane;
}
```

7.19.3.63 void KDChart::AbstractCoordinatePlane::relayout () `[inherited]`

Calling relayout() on the plane triggers the global KDChart::Chart::slotRelayout().

Definition at line 253 of file KDChartAbstractCoordinatePlane.cpp.

References KDChart::AbstractCoordinatePlane::needRelayout().

```
{
    //qDebug("KDChart::AbstractCoordinatePlane::relayout() called");
    emit needRelayout();
}
```

7.19.3.64 void KDChart::AbstractLayoutItem::removeFromParentLayout () `[inherited]`

Definition at line 78 of file KDChartLayoutItems.h.

Referenced by KDChart::Chart::takeCoordinatePlane().

```
        {
            if( mParentLayout ){
                if( widget() )
                    mParentLayout->removeWidget( widget() );
                else
                    mParentLayout->removeItem( this );
            }
        }
```

7.19.3.65 void AbstractCoordinatePlane::replaceDiagram (AbstractDiagram ∗ *diagram*, AbstractDiagram ∗ *oldDiagram* = 0) `[virtual, inherited]`

Replaces the old diagram, or appends the diagram, it there is none yet.

Parameters:

diagram The diagram to be used instead of the old diagram. This parameter must not be zero, or the method will do nothing.

oldDiagram The diagram to be removed by the new diagram. This diagram will be deleted automatically. If the parameter is omitted, the very first diagram will be replaced. In case, there was no diagram yet, the new diagram will just be added.

Note:

If you want to re-use the old diagram, call takeDiagram and addDiagram, instead of using replaceDiagram.

See also:

addDiagram, takeDiagram

Definition at line 82 of file KDChartAbstractCoordinatePlane.cpp.

References KDChart::AbstractCoordinatePlane::addDiagram(), d, KDChart::AbstractCoordinatePlane::layoutDiagrams(), KDChart::AbstractCoordinatePlane::layoutPlanes(), and KDChart::AbstractCoordinatePlane::takeDiagram().

```
{
    if( diagram && oldDiagram_ != diagram ){
        AbstractDiagram* oldDiagram = oldDiagram_;
        if( d->diagrams.count() ){
            if( ! oldDiagram )
                oldDiagram = d->diagrams.first();
            takeDiagram( oldDiagram );
        }
        delete oldDiagram;
        addDiagram( diagram );
        layoutDiagrams();
        layoutPlanes(); // there might be new axes, etc
        update();
    }
}
```

7.19.3.66 void KDChart::CartesianCoordinatePlane::resetGridAttributes (Qt::Orientation *orientation*)

Reset the attributes to be used for grid lines drawn in horizontal direction (or in vertical direction, resp.).

By calling this method you specify that the global attributes set by AbstractCoordinatePlane::setGlobalGridAttributes be used.

See also:

setGridAttributes, gridAttributes
setAutoAdjustGridToZoom
AbstractCoordinatePlane::globalGridAttributes
hasOwnGridAttributes

Definition at line 643 of file KDChartCartesianCoordinatePlane.cpp.

```
{
    setHasOwnGridAttributes( orientation, false );
    update();
}
```

7.19.3.67 int AbstractArea::rightOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the right edge of the area.

Note:

The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 85 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
    // Re-calculate the sizes,
    // so we also get the amountOf..Overlap members set newly:
    if( ! doNotRecalculate )
        sizeHint();
    return d->amountOfRightOverlap;
}
```

7.19.3.68 void KDChart::CartesianCoordinatePlane::setAutoAdjustGridToZoom (bool *autoAdjust*)

Disable / re-enable the built-in grid adjusting feature.

By default additional lines will be drawn in a Linear grid when zooming in.

See also:

autoAdjustGridToZoom, setGridAttributes

Definition at line 682 of file KDChartCartesianCoordinatePlane.cpp.

References d, and KDChart::AbstractCoordinatePlane::propertiesChanged().

```
{
    if( d->autoAdjustGridToZoom != autoAdjust ){
        d->autoAdjustGridToZoom = autoAdjust;
        d->grid->setNeedRecalculate();
        emit propertiesChanged();
    }
}
```

7.19.3.69 void CartesianCoordinatePlane::setAutoAdjustHorizontalRangeToData (unsigned int *percentEmpty* = 67)

Automatically adjust horizontal range settings to the ranges covered by the model's values, when ever the data have changed, and then emit horizontalRangeAutomaticallyAdjusted.

By default the horizontal range is adjusted automatically, if more than 67 percent of the available horizontal space would be empty otherwise.

Range setting is adjusted if more than `percentEmpty` percent of the horizontal space covered by the coordinate plane would otherwise be empty. Automatic range adjusting can happen, when either all of the data are positive or all are negative.

Set percentEmpty to 100 to disable automatic range adjusting.

Parameters:

percentEmpty The maximal percentage of horizontal space that may be empty.

See also:

horizontalRangeAutomaticallyAdjusted
autoAdjustHorizontalRangeToData, adjustRangesToData
setHorizontalRange, setVerticalRange
setAutoAdjustVerticalRangeToData

Definition at line 601 of file KDChartCartesianCoordinatePlane.cpp.

References d, layoutDiagrams(), and KDChart::AbstractCoordinatePlane::propertiesChanged().

```
{
    d->autoAdjustHorizontalRangeToData = percentEmpty;
    d->horizontalMin = 0.0;
    d->horizontalMax = 0.0;
    layoutDiagrams();
    emit propertiesChanged();
}
```

7.19.3.70 void CartesianCoordinatePlane::setAutoAdjustVerticalRangeToData (unsigned int *percentEmpty* = 67)

Automatically adjust vertical range settings to the ranges covered by the model's values, when ever the data have changed, and then emit verticalRangeAutomaticallyAdjusted.

By default the vertical range is adjusted automatically, if more than 67 percent of the available vertical space would be empty otherwise.

Range setting is adjusted if more than `percentEmpty` percent of the horizontal space covered by the coordinate plane would otherwise be empty. Automatic range adjusting can happen, when either all of the data are positive or all are negative.

Set percentEmpty to 100 to disable automatic range adjusting.

Parameters:

percentEmpty The maximal percentage of horizontal space that may be empty.

See also:

verticalRangeAutomaticallyAdjusted
autoAdjustVerticalRangeToData, adjustRangesToData
setHorizontalRange, setVerticalRange
setAutoAdjustHorizontalRangeToData

Definition at line 610 of file KDChartCartesianCoordinatePlane.cpp.

References d, layoutDiagrams(), and KDChart::AbstractCoordinatePlane::propertiesChanged().

```
{
    d->autoAdjustVerticalRangeToData = percentEmpty;
    d->verticalMin = 0.0;
    d->verticalMax = 0.0;
    layoutDiagrams();
    emit propertiesChanged();
}
```

7.19.3.71 void CartesianCoordinatePlane::setAxesCalcModes (AxesCalcMode *mode*)

Specifies the calculation modes for all axes.

Definition at line 513 of file KDChartCartesianCoordinatePlane.cpp.

References d, and KDChart::AbstractCoordinatePlane::propertiesChanged().

```
{
    if( d->coordinateTransformation.axesCalcModeY != mode ||
        d->coordinateTransformation.axesCalcModeX != mode ){
        d->coordinateTransformation.axesCalcModeY = mode;
        d->coordinateTransformation.axesCalcModeX = mode;
        emit propertiesChanged();
    }
}
```

7.19.3.72 void CartesianCoordinatePlane::setAxesCalcModeX (AxesCalcMode *mode*)

Specifies the calculation mode for all Abscissa axes.

Definition at line 531 of file KDChartCartesianCoordinatePlane.cpp.

References d, and KDChart::AbstractCoordinatePlane::propertiesChanged().

```
{
    if( d->coordinateTransformation.axesCalcModeX != mode ){
        d->coordinateTransformation.axesCalcModeX = mode;
        emit propertiesChanged();
    }
}
```

7.19.3.73 void CartesianCoordinatePlane::setAxesCalcModeY (AxesCalcMode *mode*)

Specifies the calculation mode for all Ordinate axes.

Definition at line 523 of file KDChartCartesianCoordinatePlane.cpp.

References d, and KDChart::AbstractCoordinatePlane::propertiesChanged().

```
{
    if( d->coordinateTransformation.axesCalcModeY != mode ){
        d->coordinateTransformation.axesCalcModeY = mode;
        emit propertiesChanged();
    }
}
```

7.19.3.74 void AbstractAreaBase::setBackgroundAttributes (const BackgroundAttributes & *a*) `[inherited]`

Definition at line 107 of file KDChartAbstractAreaBase.cpp.

References d.

```
{
    d->backgroundAttributes = a;
}
```

7.19.3.75 void AbstractAreaBase::setFrameAttributes (const FrameAttributes & *a*) `[inherited]`

Definition at line 97 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone().

```
{
    d->frameAttributes = a;
}
```

7.19.3.76 void KDChart::AbstractCoordinatePlane::setGeometry (const QRect & *r*) `[virtual, inherited]`

pure virtual in QLayoutItem

Note:

> Do not call this function directly, unless you know exactly what you are doing. Geometry management is done by KD Chart's internal layouting measures.

Definition at line 232 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
{
//    qDebug() << "KDChart::AbstractCoordinatePlane::setGeometry(" << r << ") called";
    if( d->geometry != r ){
        d->geometry = r;
        // Note: We do *not* call update() here
        //       because it would invoke KDChart::update() recursively.
    }
}
```

7.19.3.77 void KDChart::AbstractCoordinatePlane::setGlobalGridAttributes (const GridAttributes &) `[inherited]`

Set the grid attributes to be used by this coordinate plane.

To disable grid painting, for example, your code should like this:

```
GridAttributes ga = plane->globalGridAttributes();
ga.setGlobalGridVisible( false );
plane->setGlobalGridAttributes( ga );
```

See also:

globalGridAttributes
CartesianCoordinatePlane::setGridAttributes

Definition at line 151 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
{
    d->gridAttributes = a;
    update();
}
```

7.19.3.78 void KDChart::CartesianCoordinatePlane::setGridAttributes (Qt::Orientation *orientation*, const GridAttributes &)

Set the attributes to be used for grid lines drawn in horizontal direction (or in vertical direction, resp.).

To disable horizontal grid painting, for example, your code should like this:

```
GridAttributes ga = plane->gridAttributes( Qt::Horizontal );
ga.setGridVisible( false );
plane-setGridAttributes( Qt::Horizontal, ga );
```

Note:

setGridAttributes overwrites the global attributes that were set by AbstractCoordinatePlane::setGlobalGridAttributes. To re-activate these global attributes you can call resetGridAttributes.

See also:

resetGridAttributes, gridAttributes
setAutoAdjustGridToZoom
AbstractCoordinatePlane::setGlobalGridAttributes
hasOwnGridAttributes

Definition at line 630 of file KDChartCartesianCoordinatePlane.cpp.

References d, and KDChart::AbstractCoordinatePlane::propertiesChanged().

```
{
    if( orientation == Qt::Horizontal )
        d->gridAttributesHorizontal = a;
    else
        d->gridAttributesVertical = a;
    setHasOwnGridAttributes( orientation, true );
    update();
    emit propertiesChanged();
}
```

7.19.3.79 void KDChart::AbstractCoordinatePlane::setGridNeedsRecalculate () `[inherited]`

Used by the chart to clear the cached grid data.

Definition at line 170 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::Chart::resizeEvent().

```
{
    d->grid->setNeedRecalculate();
}
```

7.19.3.80 void KDChart::CartesianCoordinatePlane::setHorizontalRange (const QPair< qreal, qreal > & *range*)

Set the boundaries of the visible value space displayed in horizontal direction.

This is also known as the horizontal viewport.

By default the horizontal range is adjusted to the range covered by the model's data, see setAutoAdjustHorizontalRangeToData for details. Calling setHorizontalRange with a valid range disables this default automatic adjusting, while on the other hand automatic adjusting will set these ranges.

To disable use of this range you can either pass an empty pair by using the default constructor QPair() or you can set setting both values to the same which constitutes a null range.

Note:

By default the visible data range often is larger than the range calculated from the data model (or set by setHoriz.|Vert.Range(), resp.). This is due to the built-in grid calculation feature: The visible start/end values get adjusted so that they match a main-grid line. You can turn this feature off for any of the four bounds by calling GridAttributes::setAdjustBoundsToGrid() for either the global grid-attributes or for the horizontal/vertical attrs separately.

Parameters:

range a pair of values representing the smalles and the largest horizontal value space coordinate displayed.

See also:

setAutoAdjustHorizontalRangeToData, setVerticalRange
GridAttributes::setAdjustBoundsToGrid()

Definition at line 539 of file KDChartCartesianCoordinatePlane.cpp.

References d, layoutDiagrams(), and KDChart::AbstractCoordinatePlane::propertiesChanged().

```
{
    if ( d->horizontalMin != range.first || d->horizontalMax != range.second ) {
        d->autoAdjustHorizontalRangeToData = 100;
        d->horizontalMin = range.first;
        d->horizontalMax = range.second;
        layoutDiagrams();
        emit propertiesChanged();
    }
}
```

7.19.3.81 void CartesianCoordinatePlane::setIsometricScaling (bool *onOff*)

Definition at line 418 of file KDChartCartesianCoordinatePlane.cpp.

References d, layoutDiagrams(), and KDChart::AbstractCoordinatePlane::propertiesChanged().

```
{
    if ( d->isometricScaling != onOff )
    {
        d->isometricScaling = onOff;
        layoutDiagrams();
        emit propertiesChanged();
    }
}
```

7.19.3.82 void KDChart::AbstractCoordinatePlane::setParent (Chart ∗ *parent*) `[inherited]`

Called internally by KDChart::Chart.

Definition at line 185 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::Chart::addCoordinatePlane(), and KDChart::Chart::takeCoordinatePlane().

```
{
    d->parent = parent;
}
```

7.19.3.83 void KDChart::AbstractLayoutItem::setParentLayout (QLayout ∗ *lay*) `[inherited]`

Definition at line 70 of file KDChartLayoutItems.h.

```
        {
            mParentLayout = lay;
        }
```

7.19.3.84 void KDChart::AbstractLayoutItem::setParentWidget (QWidget ∗ *widget*) `[virtual, inherited]`

Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

Thus, you need to call setParentWidget on every item, that has a non-fixed size.

Definition at line 64 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::mParent.

Referenced by KDChart::Legend::buildLegend(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
{
    mParent = widget;
}
```

7.19.3.85 void KDChart::AbstractCoordinatePlane::setReferenceCoordinatePlane (AbstractCoordinatePlane ∗ *plane*) `[inherited]`

Set another coordinate plane to be used as the reference plane for this one.

Parameters:

plane The coordinate plane to be used the reference plane for this one.

See also:

referenceCoordinatePlane

Definition at line 175 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
{
    d->referenceCoordinatePlane = plane;
}
```

7.19.3.86 void KDChart::CartesianCoordinatePlane::setVerticalRange (const QPair< qreal, qreal > & *range*)

Set the boundaries of the visible value space displayed in vertical direction.

This is also known as the vertical viewport.

By default the vertical range is adjusted to the range covered by the model's data, see setAutoAdjustVerticalRangeToData for details. Calling setVerticalRange with a valid range disables this default automatic adjusting, while on the other hand automatic adjusting will set these ranges.

To disable use of this range you can either pass an empty pair by using the default constructor QPair() or you can set setting both values to the same which constitutes a null range.

Note:

By default the visible data range often is larger than the range calculated from the data model (or set by setHoriz.|Vert.Range(), resp.). This is due to the built-in grid calculation feature: The visible start/end values get adjusted so that they match a main-grid line. You can turn this feature off for any of the four bounds by calling GridAttributes::setAdjustBoundsToGrid() for either the global grid-attributes or for the horizontal/vertical attrs separately.

Parameters:

range a pair of values representing the smalles and the largest vertical value space coordinate displayed.

See also:

setAutoAdjustVerticalRangeToData, setHorizontalRange
GridAttributes::setAdjustBoundsToGrid()

Definition at line 550 of file KDChartCartesianCoordinatePlane.cpp.

References d, layoutDiagrams(), and KDChart::AbstractCoordinatePlane::propertiesChanged().

```
{

    if ( d->verticalMin != range.first || d->verticalMax != range.second ) {
        d->autoAdjustVerticalRangeToData = 100;
        d->verticalMin = range.first;
        d->verticalMax = range.second;
        layoutDiagrams();
        emit propertiesChanged();
    }
}
```

7.19.3.87 void CartesianCoordinatePlane::setZoomCenter (QPointF *center*) `[virtual]`

See also:

zoomCenter, setZoomFactorX, setZoomFactorY

Reimplemented from KDChart::AbstractCoordinatePlane.

Definition at line 480 of file KDChartCartesianCoordinatePlane.cpp.

References doneSetZoomCenter(), and KDChart::AbstractCoordinatePlane::propertiesChanged().

```
{
    if( doneSetZoomCenter( point ) ){
        emit propertiesChanged();
    }
}
```

7.19.3.88 void CartesianCoordinatePlane::setZoomFactorX (double *factor*) `[virtual]`

See also:

zoomFactorX, setZoomCenter

Reimplemented from KDChart::AbstractCoordinatePlane.

Definition at line 466 of file KDChartCartesianCoordinatePlane.cpp.

References doneSetZoomFactorX(), and KDChart::AbstractCoordinatePlane::propertiesChanged().

```
{
    if( doneSetZoomFactorX( factor ) ){
        emit propertiesChanged();
    }
}
```

7.19.3.89 void CartesianCoordinatePlane::setZoomFactorY (double *factor*) `[virtual]`

See also:

zoomFactorY, setZoomCenter

Reimplemented from KDChart::AbstractCoordinatePlane.

Definition at line 473 of file KDChartCartesianCoordinatePlane.cpp.

References doneSetZoomFactorY(), and KDChart::AbstractCoordinatePlane::propertiesChanged().

```
{
    if( doneSetZoomFactorY( factor ) ){
        emit propertiesChanged();
    }
}
```

7.19.3.90 QSize KDChart::AbstractCoordinatePlane::sizeHint () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 225 of file KDChartAbstractCoordinatePlane.cpp.

References KDChart::AbstractCoordinatePlane::maximumSize().

```
{
    // we return our maxiumu (which is the full size of the Chart)
    // even if we know the plane will be smaller
    return maximumSize();
}
```

7.19.3.91 void KDChart::AbstractLayoutItem::sizeHintChanged () const `[virtual, inherited]`

Report changed size hint: ask the parent widget to recalculate the layout.

Definition at line 86 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextLayoutItem::sizeHint().

```
{
    // This is exactly like what QWidget::updateGeometry does.
//  qDebug("KDChart::AbstractLayoutItem::sizeHintChanged() called");
    if( mParent ) {
        if ( mParent->layout() )
            mParent->layout()->invalidate();
        else
            QApplication::postEvent( mParent, new QEvent( QEvent::LayoutRequest ) );
    }
}
```

7.19.3.92 QSizePolicy KDChart::AbstractCoordinatePlane::sizePolicy () const `[virtual, inherited]`

[reimplemented]

Definition at line 146 of file KDChartAbstractCoordinatePlane.cpp.

```
{
    return QSizePolicy( QSizePolicy::MinimumExpanding, QSizePolicy::MinimumExpanding );
}
```

7.19.3.93 void AbstractCoordinatePlane::takeDiagram (AbstractDiagram ∗ *diagram*) `[virtual, inherited]`

Removes the diagram from the plane, without deleting it.

The plane no longer owns the diagram, so it is the caller's responsibility to delete the diagram.

See also:

addDiagram, replaceDiagram

Definition at line 100 of file KDChartAbstractCoordinatePlane.cpp.

References d, KDChart::AbstractCoordinatePlane::layoutDiagrams(), and KDChart::AbstractDiagram::setCoordinatePlane().

Referenced by KDChart::AbstractCoordinatePlane::replaceDiagram().

```
{
    const int idx = d->diagrams.indexOf( diagram );
    if( idx != -1 ){
        d->diagrams.removeAt( idx );
        diagram->setParent( 0 );
        diagram->setCoordinatePlane( 0 );
        layoutDiagrams();
        update();
    }
}
```

7.19.3.94 int AbstractArea::topOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the top edge of the area.

Note:

The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 93 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
    // Re-calculate the sizes,
    // so we also get the amountOf..Overlap members set newly:
    if( ! doNotRecalculate )
        sizeHint();
    return d->amountOfTopOverlap;
}
```

7.19.3.95 const QPointF CartesianCoordinatePlane::translate (const QPointF & *diagramPoint*) const `[virtual]`

Translate the given point in value space coordinates to a position in pixel space.

Parameters:

diagramPoint The point in value coordinates.

Returns:

The translated point.

Implements KDChart::AbstractCoordinatePlane.

Definition at line 404 of file KDChartCartesianCoordinatePlane.cpp.

References d.

Referenced by KDChart::BarDiagram::paint(), and KDChart::CartesianAxis::paintCtx().

```
{
    // Note: We do not test if the point lays inside of the data area,
    //       but we just apply the transformation calculations to the point.
    //       This allows for basic calculations done by the user, see e.g.
    //       the file  examples/Lines/BubbleChart/mainwindow.cpp
    return  d->coordinateTransformation.translate ( diagramPoint );
}
```

7.19.3.96 const QPointF CartesianCoordinatePlane::translateBack (const QPointF & *screenPoint*) const `[protected]`

Definition at line 413 of file KDChartCartesianCoordinatePlane.cpp.

References d.

```
{
    return  d->coordinateTransformation.translateBack ( screenPoint );
}
```

7.19.3.97 QPair< qreal, qreal > KDChart::CartesianCoordinatePlane::verticalRange () const

Returns:

The largest and smallest visible horizontal value space value. If this is not explicitely set, or if both values are the same, the plane will use the union of the dataBoundaries of all associated diagrams.

See also:

KDChart::AbstractDiagram::dataBoundaries

Definition at line 567 of file KDChartCartesianCoordinatePlane.cpp.

References d.

```
{
    return QPair<qreal, qreal>( d->verticalMin, d->verticalMax );
}
```

7.19.3.98 QPointF CartesianCoordinatePlane::zoomCenter () const `[virtual]`

See also:
setZoomCenter, setZoomFactorX, setZoomFactorY

Reimplemented from KDChart::AbstractCoordinatePlane.

Definition at line 487 of file KDChartCartesianCoordinatePlane.cpp.

References d.

```
{
    return d->coordinateTransformation.zoom.center();
}
```

7.19.3.99 double CartesianCoordinatePlane::zoomFactorX () const `[virtual]`

See also:
setZoomFactorX, setZoomCenter

Reimplemented from KDChart::AbstractCoordinatePlane.

Definition at line 492 of file KDChartCartesianCoordinatePlane.cpp.

References d.

```
{
    return d->coordinateTransformation.zoom.xFactor;
}
```

7.19.3.100 double CartesianCoordinatePlane::zoomFactorY () const `[virtual]`

See also:
setZoomFactorY, setZoomCenter

Reimplemented from KDChart::AbstractCoordinatePlane.

Definition at line 497 of file KDChartCartesianCoordinatePlane.cpp.

References d.

```
{
    return d->coordinateTransformation.zoom.yFactor;
}
```

7.19.4 Member Data Documentation

7.19.4.1 Q_SIGNALS KDChart::AbstractCoordinatePlane::__pad0__ `[inherited]`

Reimplemented from KDChart::AbstractArea.

Definition at line 297 of file KDChartAbstractCoordinatePlane.h.

7.19.4.2 QWidget∗ KDChart::AbstractLayoutItem::mParent `[protected, inherited]`

Definition at line 88 of file KDChartLayoutItems.h.

Referenced by KDChart::AbstractLayoutItem::setParentWidget().

7.19.4.3 QLayout∗ KDChart::AbstractLayoutItem::mParentLayout `[protected, inherited]`

Definition at line 89 of file KDChartLayoutItems.h.

7.19.4.4 protected KDChart::CartesianCoordinatePlane::Q_SLOTS `[protected]`

Reimplemented from KDChart::AbstractCoordinatePlane.

Definition at line 374 of file KDChartCartesianCoordinatePlane.h.

7.19.4.5 public KDChart::CartesianCoordinatePlane::Q_SLOTS

Reimplemented from KDChart::AbstractCoordinatePlane.

Definition at line 342 of file KDChartCartesianCoordinatePlane.h.

The documentation for this class was generated from the following files:

- KDChartCartesianCoordinatePlane.h
- KDChartCartesianCoordinatePlane.cpp

7.20 KDChart::Chart Class Reference

`#include <KDChartChart>`

Inheritance diagram for KDChart::Chart:Collaboration diagram for KDChart::Chart:

7.20.1 Detailed Description

A chart with one or more diagrams.

The Chart class represents a drawing consisting of one or more diagrams and various optional elements such as legends, axes, text boxes, headers or footers. It takes ownership of all these elements when they are assigned to it. Each diagram is associated with a coordinate plane, of which the chart can have more than one. The coordinate planes (and thus the associated diagrams) can be layed out in various ways.

The Chart class makes heavy use of the Qt Interview framework for model/view programming, and thus requires data to be presented to it in a QAbstractItemModel compatible way. For many simple charts, especially if the visualized data is static, KDChart::Widget provides an abstracted interface, that hides the complexity of Interview to a large extent.

Definition at line 72 of file KDChartChart.h.

Public Member Functions

- void addCoordinatePlane (AbstractCoordinatePlane ∗plane)

  Adds a coordinate plane to the chart.

- void addHeaderFooter (HeaderFooter ∗headerFooter)

  Adds a header or a footer to the chart.

- void addLegend (Legend ∗legend)

  Add the given legend to the chart.

- BackgroundAttributes backgroundAttributes () const
- Chart (QWidget ∗parent=0)
- AbstractCoordinatePlane ∗ coordinatePlane ()

  Each chart must have at least one coordinate plane.

- QLayout ∗ coordinatePlaneLayout ()
- CoordinatePlaneList coordinatePlanes ()

  The list of coordinate planes.

- FrameAttributes frameAttributes () const
- int globalLeadingBottom () const

  The padding between the start of the widget and the start of the area that is used for drawing at the bottom.

- int globalLeadingLeft () const

  The padding between the start of the widget and the start of the area that is used for drawing on the left.

- int globalLeadingRight () const

  The padding between the start of the widget and the start of the area that is used for drawing on the right.

- int globalLeadingTop () const

  The padding between the start of the widget and the start of the area that is used for drawing at the top.

- HeaderFooter ∗ headerFooter ()

  The first header or footer of the chart.

- HeaderFooterList headerFooters ()

  The list of headers and footers associated with the chart.

- Legend ∗ legend ()

  The first legend of the chart or 0 if there was none added to the chart.

- LegendList legends ()

  The list of all legends associated with the chart.

- void paint (QPainter ∗painter, const QRect &target)

  Paints all the contents of the chart.

- void reLayoutFloatingLegends ()
- void replaceCoordinatePlane (AbstractCoordinatePlane ∗plane, AbstractCoordinatePlane ∗oldPlane=0)

  Replaces the old coordinate plane, or appends the plane, it there is none yet.

- void replaceHeaderFooter (HeaderFooter ∗headerFooter, HeaderFooter ∗oldHeaderFooter=0)

  Replaces the old header (or footer, resp.), or appends the new header or footer, it there is none yet.

- void replaceLegend (Legend ∗legend, Legend ∗oldLegend=0)

  Replaces the old legend, or appends the new legend, it there is none yet.

- void setBackgroundAttributes (const BackgroundAttributes &a)

  Specify the background attributes to be used, by default there is no background.

- void setCoordinatePlaneLayout (QLayout ∗layout)
- void setFrameAttributes (const FrameAttributes &a)

  Specify the frame attributes to be used, by default is it a thin black line.

- void setGlobalLeading (int left, int top, int right, int bottom)

  Set the padding between the margin of the widget and the area that the contents are drawn into.

- void setGlobalLeadingBottom (int leading)

  Set the padding between the start of the widget and the start of the area that is used for drawing on the bottom.

- void setGlobalLeadingLeft (int leading)

  Set the padding between the start of the widget and the start of the area that is used for drawing on the left.

- void setGlobalLeadingRight (int leading)

  Set the padding between the start of the widget and the start of the area that is used for drawing on the right.

- void setGlobalLeadingTop (int leading)

Set the padding between the start of the widget and the start of the area that is used for drawing at the top.

- void takeCoordinatePlane (AbstractCoordinatePlane ∗plane)

  Removes the coordinate plane from the chart, without deleting it.

- void takeHeaderFooter (HeaderFooter ∗headerFooter)

  Removes the header (or footer, resp.) from the chart, without deleting it.

- void takeLegend (Legend ∗legend)

  Removes the legend from the chart, without deleting it.

- ∼Chart ()

Public Attributes

- Q_SIGNALS __pad0__: void propertiesChanged()

Protected Member Functions

- void mousePressEvent (QMouseEvent ∗event)

  reimp

- void paintEvent (QPaintEvent ∗event)

  Draws the background and frame, then calls paint().

- void resizeEvent (QResizeEvent ∗event)

  Adjusts the internal layout when the chart is resized.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 Chart::Chart (QWidget ∗ *parent* = 0) `[explicit]`

Definition at line 781 of file KDChartChart.cpp.

References addCoordinatePlane(), setFrameAttributes(), KDChart::FrameAttributes::setPadding(), KDChart::FrameAttributes::setPen(), and KDChart::FrameAttributes::setVisible().

```
    : QWidget ( parent )
    , _d( new Private( this ) )
{
#if defined KDAB_EVAL
    EvalDialog::checkEvalLicense( "KD Chart" );
#endif

    FrameAttributes frameAttrs;
    frameAttrs.setVisible( true );
    frameAttrs.setPen( QPen( Qt::black ) );
    frameAttrs.setPadding( 1 );
    setFrameAttributes( frameAttrs );

    addCoordinatePlane( new CartesianCoordinatePlane ( this ) );
}
```

7.20.2.2 Chart::∼Chart ()

Definition at line 798 of file KDChartChart.cpp.

```
799 {
800     delete _d;
801 }
```

7.20.3 Member Function Documentation

7.20.3.1 void Chart::addCoordinatePlane (AbstractCoordinatePlane ∗ *plane*)

Adds a coordinate plane to the chart.

The chart takes ownership.

Parameters:

plane The coordinate plane to add.

See also:

replaceCoordinatePlane, takeCoordinatePlane

Definition at line 846 of file KDChartChart.cpp.

References d, and KDChart::AbstractCoordinatePlane::setParent().

Referenced by Chart(), and replaceCoordinatePlane().

```
847 {
848     connect( plane, SIGNAL( destroyedCoordinatePlane( AbstractCoordinatePlane* ) ),
849              d,   SLOT( slotUnregisterDestroyedPlane( AbstractCoordinatePlane* ) ) );
850     connect( plane, SIGNAL( needUpdate() ),       this,   SLOT( update() ) );
851     connect( plane, SIGNAL( needRelayout() ),     d,      SLOT( slotRelayout() ) ) ;
852     connect( plane, SIGNAL( needLayoutPlanes() ), d,      SLOT( slotLayoutPlanes() ) ) ;
853     connect( plane, SIGNAL( propertiesChanged() ),this, SIGNAL( propertiesChanged() ) );
854     d->coordinatePlanes.append( plane );
855     plane->setParent( this );
856     d->slotLayoutPlanes();
857 }
```

7.20.3.2 void Chart::addHeaderFooter (HeaderFooter ∗ *headerFooter*)

Adds a header or a footer to the chart.

The chart takes ownership.

Parameters:

headerFooter The header (or footer, resp.) to add.

See also:

replaceHeaderFooter, takeHeaderFooter

Definition at line 1036 of file KDChartChart.cpp.

References d, and KDChart::HeaderFooter::setParent().

Referenced by replaceHeaderFooter().

```
{
    d->headerFooters.append( headerFooter );
    headerFooter->setParent( this );
    connect( headerFooter, SIGNAL( destroyedHeaderFooter( HeaderFooter* ) ),
             d, SLOT( slotUnregisterDestroyedHeaderFooter( HeaderFooter* ) ) );
    connect( headerFooter, SIGNAL( positionChanged( HeaderFooter* ) ),
             d, SLOT( slotRelayout() ) );
    d->slotRelayout();
}
```

7.20.3.3 void Chart::addLegend (Legend ∗ *legend*)

Add the given legend to the chart.

The chart takes ownership.

Parameters:

legend The legend to add.

See also:

replaceLegend, takeLegend

Definition at line 1089 of file KDChartChart.cpp.

References d, KDChart::TextAttributes::fontSize(), KDChart::TextAttributes::setFontSize(), KDChart::Legend::setReferenceArea(), KDChart::Measure::setRelativeMode(), KDChart::Legend::setTextAttributes(), KDChart::Legend::setTitleTextAttributes(), KDChart::Measure::setValue(), KDChart::Legend::setVisible(), KDChart::Legend::textAttributes(), and KDChart::Legend::titleTextAttributes().

Referenced by replaceLegend().

```
{
    if( ! legend ) return;

    //qDebug() << "adding the legend";
    d->legends.append( legend );
    legend->setParent( this );

    TextAttributes textAttrs( legend->textAttributes() );

    KDChart::Measure measure( textAttrs.fontSize() );
    measure.setRelativeMode( this, KDChartEnums::MeasureOrientationMinimum );
    measure.setValue( 20 );
    textAttrs.setFontSize( measure );
    legend->setTextAttributes( textAttrs );

    textAttrs = legend->titleTextAttributes();
    measure.setRelativeMode( this, KDChartEnums::MeasureOrientationMinimum );
    measure.setValue( 24 );
    textAttrs.setFontSize( measure );

    legend->setTitleTextAttributes( textAttrs );

    legend->setReferenceArea( this );

/*
    future: Use relative sizes for the markers too!

    const uint nMA = Legend::datasetCount();
    for( uint iMA = 0; iMA < nMA; ++iMA ){
        MarkerAttributes ma( legend->markerAttributes( iMA ) );
        ma.setMarkerSize( ... )
        legend->setMarkerAttributes( iMA, ma )
    }
*/

    connect( legend, SIGNAL( destroyedLegend( Legend* ) ),
             d, SLOT( slotUnregisterDestroyedLegend( Legend* ) ) );
    connect( legend, SIGNAL( positionChanged( AbstractAreaWidget* ) ),
             d, SLOT( slotLayoutPlanes() ) ); //slotRelayout() ) );
    connect( legend, SIGNAL( propertiesChanged() ),this, SIGNAL( propertiesChanged() ) );
    legend->setVisible( true );
    d->slotRelayout();
}
```

7.20.3.4 BackgroundAttributes Chart::backgroundAttributes () const

Definition at line 820 of file KDChartChart.cpp.

References d.

```
{
    return d->backgroundAttributes;
}
```

7.20.3.5 AbstractCoordinatePlane ∗ Chart::coordinatePlane ()

Each chart must have at least one coordinate plane.

Initially a default CartesianCoordinatePlane is created. Use replaceCoordinatePlane() to replace it with a different one, such as a PolarCoordinatePlane.

Returns:
The first coordinate plane of the chart.

Definition at line 830 of file KDChartChart.cpp.

References d.

```
{
    if ( d->coordinatePlanes.isEmpty() )
    {
        qWarning() << "Chart::coordinatePlane: warning: no coordinate plane defined.";
        return 0;
    } else {
        return d->coordinatePlanes.first();
    }
}
```

7.20.3.6 QLayout ∗ Chart::coordinatePlaneLayout ()

Definition at line 825 of file KDChartChart.cpp.

References d.

```
{
    return d->planesLayout;
}
```

7.20.3.7 CoordinatePlaneList Chart::coordinatePlanes ()

The list of coordinate planes.

Returns:

The list of coordinate planes.

Definition at line 841 of file KDChartChart.cpp.

References KDChart::CoordinatePlaneList, and d.

```
{
    return d->coordinatePlanes;
}
```

7.20.3.8 FrameAttributes Chart::frameAttributes () const

Definition at line 810 of file KDChartChart.cpp.

References d.

```
{
    return d->frameAttributes;
}
```

7.20.3.9 int Chart::globalLeadingBottom () const

The padding between the start of the widget and the start of the area that is used for drawing at the bottom.

Returns:

The padding between the start of the widget and the start of the area that is used for drawing at the bottom.

See also:

setGlobalLeading

Definition at line 935 of file KDChartChart.cpp.

References d.

```
{
    return d->globalLeadingBottom;
}
```

7.20.3.10 int Chart::globalLeadingLeft () const

The padding between the start of the widget and the start of the area that is used for drawing on the left.

Returns:

The padding between the start of the widget and the start of the area that is used for drawing on the left.

See also:
setGlobalLeading

Definition at line 902 of file KDChartChart.cpp.

References d.

```
{
    return d->globalLeadingLeft;
}
```

7.20.3.11 int Chart::globalLeadingRight () const

The padding between the start of the widget and the start of the area that is used for drawing on the right.

Returns:
The padding between the start of the widget and the start of the area that is used for drawing on the right.

See also:
setGlobalLeading

Definition at line 924 of file KDChartChart.cpp.

References d.

```
{
    return d->globalLeadingRight;
}
```

7.20.3.12 int Chart::globalLeadingTop () const

The padding between the start of the widget and the start of the area that is used for drawing at the top.

Returns:
The padding between the start of the widget and the start of the area that is used for drawing at the top.

See also:
setGlobalLeading

Definition at line 913 of file KDChartChart.cpp.

References d.

```
{
    return d->globalLeadingTop;
}
```

7.20.3.13 HeaderFooter ∗ Chart::headerFooter ()

The first header or footer of the chart.

By default there is none.

Returns:
The first header or footer of the chart or 0 if there was none added to the chart.

Definition at line 1074 of file KDChartChart.cpp.

References d.

```
{
    if( d->headerFooters.isEmpty() ) {
        return 0;
    } else {
        return d->headerFooters.first();
    }
}
```

7.20.3.14 HeaderFooterList Chart::headerFooters ()

The list of headers and footers associated with the chart.

Returns:
The list of headers and footers associated with the chart.

Definition at line 1083 of file KDChartChart.cpp.

References d, and KDChart::HeaderFooterList.

```
{
    return d->headerFooters;
}
```

7.20.3.15 Legend ∗ Chart::legend ()

The first legend of the chart or 0 if there was none added to the chart.

Returns:
The first legend of the chart or 0 if none exists.

Definition at line 1161 of file KDChartChart.cpp.

References d.

Referenced by paint(), and reLayoutFloatingLegends().

```
{
    if ( d->legends.isEmpty() )
    {
        return 0;
    } else {
        return d->legends.first();
    }
}
```

7.20.3.16 LegendList Chart::legends ()

The list of all legends associated with the chart.

Returns:

The list of all legends associated with the chart.

Definition at line 1171 of file KDChartChart.cpp.

References d, and KDChart::LegendList.

```
{
    return d->legends;
}
```

7.20.3.17 void Chart::mousePressEvent (QMouseEvent ∗ *event*) `[protected]`

reimp

Definition at line 1177 of file KDChartChart.cpp.

References d, KDChart::AbstractCoordinatePlane::diagram(), KDChart::AbstractCoordinatePlane::diagrams(), KDChart::AbstractCoordinatePlane::geometry(), and KDChart::AbstractCoordinatePlane::mousePressEvent().

```
{
    KDAB_FOREACH( AbstractCoordinatePlane* plane, d->coordinatePlanes ) {
       if ( plane->geometry().contains( event->pos() ) ) {
           if ( plane->diagrams().size() > 0 ) {
               QPoint pos = plane->diagram()->mapFromGlobal( event->globalPos() );
               QMouseEvent ev( QEvent::MouseButtonPress, pos, event->globalPos(),
                               event->button(), event->buttons(),
                               event->modifiers() );
               plane->mousePressEvent( &ev );
           }
       }
    }
}
```

7.20.3.18 void Chart::paint (QPainter ∗ *painter*, const QRect & *target*)

Paints all the contents of the chart.

Use this method, to make KDChart draw into your QPainter.

Note:

Any global leading settings will be used by the paint method too, so make sure to set them to zero, if you want the drawing to have the exact size of the target rectangle.

Parameters:

painter The painter to be drawn into.

target The rectangle to be filled by the Chart's drawing.

See also:

setGlobalLeading

Definition at line 940 of file KDChartChart.cpp.

References d, legend(), and KDChart::AbstractAreaWidget::paintIntoRect().

```
{
    if( target.isEmpty() || !painter ) return;
    //qDebug() << "Chart::paint( ..," << target << ")";

    GlobalMeasureScaling::instance()->setFactors(
            static_cast<qreal>(target.width()) /
            static_cast<qreal>(geometry().size().width()),
            static_cast<qreal>(target.height()) /
            static_cast<qreal>(geometry().size().height()) );

    if( target.size() != d->currentLayoutSize ){
        d->resizeLayout( target.size() );
    }
    const QPoint translation = target.topLeft();
    painter->translate( translation );

    d->paintAll( painter );

    // for debugging:
    //painter->setPen(QPen(Qt::blue, 8));
    //painter->drawRect(target.adjusted(12,12,-12,-12));

    KDAB_FOREACH( Legend *legend, d->legends ) {
        const bool hidden = legend->isHidden() && legend->testAttribute(Qt::WA_WState_ExplicitShowHide
        if ( !hidden ) {
            //qDebug() << "painting legend at " << legend->geometry();
            legend->paintIntoRect( *painter, legend->geometry() );
            //testing:
            //legend->paintIntoRect( *painter, legend->geometry().adjusted(-100,0,-100,0) );
        }
    }

    painter->translate( -translation.x(), -translation.y() );

    GlobalMeasureScaling::instance()->resetFactors();

    //qDebug() << "KDChart::Chart::paint() done.\n";
}
```

7.20.3.19 void Chart::paintEvent (QPaintEvent ∗ *event*) `[protected]`

Draws the background and frame, then calls paint().

In most cases there is no need to override this method in a derived class, but if you do, do not forget to call paint().

See also:
paint

Definition at line 1022 of file KDChartChart.cpp.

References d, and reLayoutFloatingLegends().

```
{
    QPainter painter( this );

    if( size() != d->currentLayoutSize ){
        d->resizeLayout( size() );
        reLayoutFloatingLegends();
    }

    //FIXME(khz): Paint the background/frame too!
    //            (can we derive Chart from AreaWidget ??)
    d->paintAll( &painter );
}
```

7.20.3.20 void Chart::reLayoutFloatingLegends ()

Definition at line 990 of file KDChartChart.cpp.

References KDChart::RelativePosition::alignment(), KDChart::RelativePosition::calculatedPoint(), d, KDChart::Legend::floatingPosition(), KDChart::Position::isFloating(), legend(), KDChart::Legend::position(), and KDChart::Legend::sizeHint().

Referenced by paintEvent(), and resizeEvent().

```
{
    KDAB_FOREACH( Legend *legend, d->legends ) {
        const bool hidden = legend->isHidden() && legend->testAttribute(Qt::WA_WState_ExplicitShowHide
        if ( legend->position().isFloating() && !hidden ){
            // resize the legend
            const QSize legendSize( legend->sizeHint() );
            legend->setGeometry( QRect( legend->geometry().topLeft(), legendSize ) );
            // find the legends corner point (reference point plus any paddings)
            const RelativePosition relPos( legend->floatingPosition() );
            QPointF pt( relPos.calculatedPoint( size() ) );
            qDebug() << pt;
            // calculate the legend’s top left point
            const Qt::Alignment alignTopLeft = Qt::AlignBottom | Qt::AlignLeft;
            if( (relPos.alignment() & alignTopLeft) != alignTopLeft ){
                if( relPos.alignment() & Qt::AlignRight )
                    pt.rx() -= legendSize.width();
                else if( relPos.alignment() & Qt::AlignHCenter )
                    pt.rx() -= 0.5 * legendSize.width();

                if( relPos.alignment() & Qt::AlignBottom )
                    pt.ry() -= legendSize.height();
                else if( relPos.alignment() & Qt::AlignVCenter )
                    pt.ry() -= 0.5 * legendSize.height();
            }
            qDebug() << pt << endl;
            legend->move( static_cast<int>(pt.x()), static_cast<int>(pt.y()) );
        }
    }
}
```

7.20.3.21 void Chart::replaceCoordinatePlane (AbstractCoordinatePlane ∗ *plane*, AbstractCoordinatePlane ∗ *oldPlane* = 0)

Replaces the old coordinate plane, or appends the plane, it there is none yet.

Parameters:

plane The coordinate plane to be used instead of the old plane. This parameter must not be zero, or the method will do nothing.

oldPlane The coordinate plane to be removed by the new plane. This plane will be deleted automatically. If the parameter is omitted, the very first coordinate plane will be replaced. In case, there was no plane yet, the new plane will just be added.

Note:

If you want to re-use the old coordinate plane, call takeCoordinatePlane and addCoordinatePlane, instead of using replaceCoordinatePlane.

See also:

addCoordinatePlane, takeCoordinatePlane

Definition at line 859 of file KDChartChart.cpp.

References addCoordinatePlane(), d, and takeCoordinatePlane().

```
{
    if( plane && oldPlane_ != plane ){
        AbstractCoordinatePlane* oldPlane = oldPlane_;
        if( d->coordinatePlanes.count() ){
            if( ! oldPlane )
                oldPlane = d->coordinatePlanes.first();
            takeCoordinatePlane( oldPlane );
        }
        delete oldPlane;
        addCoordinatePlane( plane );
    }
}
```

7.20.3.22 void Chart::replaceHeaderFooter (HeaderFooter ∗ *headerFooter*, HeaderFooter ∗ *oldHeaderFooter* = 0)

Replaces the old header (or footer, resp.), or appends the new header or footer, it there is none yet.

Parameters:

headerFooter The header or footer to be used instead of the old one. This parameter must not be zero, or the method will do nothing.

oldHeaderFooter The header or footer to be removed by the new one. This header or footer will be deleted automatically. If the parameter is omitted, the very first header or footer will be replaced. In case, there was no header and no footer yet, the new header or footer will just be added.

Note:

If you want to re-use the old header or footer, call takeHeaderFooter and addHeaderFooter, instead of using replaceHeaderFooter.

See also:

addHeaderFooter, takeHeaderFooter

Definition at line 1047 of file KDChartChart.cpp.

References addHeaderFooter(), d, and takeHeaderFooter().

```
{
    if( headerFooter && oldHeaderFooter_ != headerFooter ){
        HeaderFooter* oldHeaderFooter = oldHeaderFooter_;
        if( d->headerFooters.count() ){
            if( ! oldHeaderFooter )
                oldHeaderFooter =  d->headerFooters.first();
            takeHeaderFooter( oldHeaderFooter );
        }
        delete oldHeaderFooter;
        addHeaderFooter( headerFooter );
    }
}
```

7.20.3.23 void Chart::replaceLegend (Legend ∗ *legend*, Legend ∗ *oldLegend* = 0)

Replaces the old legend, or appends the new legend, it there is none yet.

Parameters:

legend The legend to be used instead of the old one. This parameter must not be zero, or the method will do nothing.

oldLegend The legend to be removed by the new one. This legend will be deleted automatically. If the parameter is omitted, the very first legend will be replaced. In case, there was no legend yet, the new legend will just be added.

If you want to re-use the old legend, call takeLegend and addLegend, instead of using replaceLegend.

Note:

Whenever addLegend is called the font sizes used by the Legend are set to relative and they get coupled to the Chart's size, with their relative values being 20 for the item texts and 24 to the title text. So if you want to use custom font sizes for the Legend make sure to set them after calling addLegend.

See also:

addLegend, takeLegend

Definition at line 1135 of file KDChartChart.cpp.

References addLegend(), d, and takeLegend().

```
{
    if( legend && oldLegend_ != legend ){
        Legend* oldLegend = oldLegend_;
        if( d->legends.count() ){
            if( ! oldLegend )
                oldLegend = d->legends.first();
            takeLegend( oldLegend );
        }
        delete oldLegend;
        addLegend( legend );
    }
}
```

7.20.3.24 void Chart::resizeEvent (QResizeEvent ∗ *event*) `[protected]`

Adjusts the internal layout when the chart is resized.

Definition at line 980 of file KDChartChart.cpp.

References d, reLayoutFloatingLegends(), and KDChart::AbstractCoordinatePlane::setGridNeedsRecalculate().

```
{
    d->resizeLayout( size() );
    KDAB_FOREACH( AbstractCoordinatePlane* plane, d->coordinatePlanes ){
        plane->setGridNeedsRecalculate();
    }
    reLayoutFloatingLegends();
}
```

7.20.3.25 void Chart::setBackgroundAttributes (const BackgroundAttributes & *a*)

Specify the background attributes to be used, by default there is no background.

To set a light blue background, you could do something like this:

```
KDChart::BackgroundAttributes backgroundAttrs( my_chart->backgroundAttributes() );
backgroundAttrs.setVisible( true );
backgroundAttrs.setBrush( QColor(0xd0,0xd0,0xff) );
my_chart->setBackgroundAttributes( backgroundAttrs );
```

See also:
setFrameAttributes

Definition at line 815 of file KDChartChart.cpp.

References d.

```
816 {
817     d->backgroundAttributes = a;
818 }
```

7.20.3.26 void KDChart::Chart::setCoordinatePlaneLayout (QLayout $*$ *layout*)

7.20.3.27 void Chart::setFrameAttributes (const FrameAttributes & *a*)

Specify the frame attributes to be used, by default is it a thin black line.

To hide the frame line, you could do something like this:

```
KDChart::FrameAttributes frameAttrs( my_chart->frameAttributes() );
frameAttrs.setVisible( false );
my_chart->setFrameAttributes( frameAttrs );
```

See also:
setBackgroundAttributes

Definition at line 805 of file KDChartChart.cpp.

References d.

Referenced by Chart().

```
806 {
807     d->frameAttributes = a;
808 }
```

7.20.3.28 void Chart::setGlobalLeading (int *left*, int *top*, int *right*, int *bottom*)

Set the padding between the margin of the widget and the area that the contents are drawn into.

Parameters:
left The padding on the left side.

top The padding at the top.

right The padding on the left hand side.

bottom The padding on the bottom.

Note:

Using previous versions of KD Chart you might have called setGlobalLeading() to make room for long Abscissa labels (or for an overlapping top label of an Ordinate axis, resp.) that would not fit into the normal axis area. This is *no longer needed* because KD Chart now is using hidden auto-spacer items reserving as much free space as is needed for axes with overlaping content at the respective sides.

See also:

setGlobalLeadingTop, setGlobalLeadingBottom, setGlobalLeadingLeft, setGlobalLeadingRight globalLeadingTop, globalLeadingBottom, globalLeadingLeft, globalLeadingRight

Definition at line 887 of file KDChartChart.cpp.

References d, setGlobalLeadingBottom(), setGlobalLeadingLeft(), setGlobalLeadingRight(), and setGlobalLeadingTop().

```
888 {
889     setGlobalLeadingLeft( left );
890     setGlobalLeadingTop( top );
891     setGlobalLeadingRight( right );
892     setGlobalLeadingBottom( bottom );
893     d->slotRelayout();
894 }
```

7.20.3.29 void Chart::setGlobalLeadingBottom (int *leading*)

Set the padding between the start of the widget and the start of the area that is used for drawing on the bottom.

Parameters:

leading The padding value.

See also:

setGlobalLeading

Definition at line 929 of file KDChartChart.cpp.

References d.

Referenced by setGlobalLeading().

```
930 {
931     d->globalLeadingBottom = leading;
932     d->slotRelayout();
933 }
```

7.20.3.30 void Chart::setGlobalLeadingLeft (int *leading*)

Set the padding between the start of the widget and the start of the area that is used for drawing on the left.

Parameters:

leading The padding value.

See also:

setGlobalLeading

Definition at line 896 of file KDChartChart.cpp.

References d.

Referenced by setGlobalLeading().

```
{
    d->globalLeadingLeft = leading;
    d->slotRelayout();
}
```

7.20.3.31 void Chart::setGlobalLeadingRight (int *leading*)

Set the padding between the start of the widget and the start of the area that is used for drawing on the right.

Parameters:

leading The padding value.

See also:

setGlobalLeading

Definition at line 918 of file KDChartChart.cpp.

References d.

Referenced by setGlobalLeading().

```
{
    d->globalLeadingRight = leading;
    d->slotRelayout();
}
```

7.20.3.32 void Chart::setGlobalLeadingTop (int *leading*)

Set the padding between the start of the widget and the start of the area that is used for drawing at the top.

Parameters:

leading The padding value.

See also:

setGlobalLeading

Definition at line 907 of file KDChartChart.cpp.

References d.

Referenced by setGlobalLeading().

```
{
    d->globalLeadingTop = leading;
    d->slotRelayout();
}
```

7.20.3.33 void Chart::takeCoordinatePlane (AbstractCoordinatePlane ∗ *plane*)

Removes the coordinate plane from the chart, without deleting it.

The chart no longer owns the plane, so it is the caller's responsibility to delete the plane.

See also:
addCoordinatePlane, takeCoordinatePlane

Definition at line 874 of file KDChartChart.cpp.

References d, KDChart::AbstractLayoutItem::removeFromParentLayout(), and KDChart::AbstractCoordinatePlane::setParent().

Referenced by replaceCoordinatePlane().

```
{
    const int idx = d->coordinatePlanes.indexOf( plane );
    if( idx != -1 ){
        d->coordinatePlanes.takeAt( idx );
        disconnect( plane, SIGNAL( destroyedCoordinatePlane( AbstractCoordinatePlane* ) ),
                    d, SLOT( slotUnregisterDestroyedPlane( AbstractCoordinatePlane* ) ) );
        plane->removeFromParentLayout();
        plane->setParent( 0 );
    }
    d->slotLayoutPlanes();
}
```

7.20.3.34 void Chart::takeHeaderFooter (HeaderFooter ∗ *headerFooter*)

Removes the header (or footer, resp.) from the chart, without deleting it.

The chart no longer owns the header or footer, so it is the caller's responsibility to delete the header or footer.

See also:
addHeaderFooter, replaceHeaderFooter

Definition at line 1062 of file KDChartChart.cpp.

References d, and KDChart::HeaderFooter::setParent().

Referenced by replaceHeaderFooter().

```
{
    const int idx = d->headerFooters.indexOf( headerFooter );
    if( idx != -1 ){
        d->headerFooters.takeAt( idx );
        disconnect( headerFooter, SIGNAL( destroyedHeaderFooter( HeaderFooter* ) ),
                    d, SLOT( slotUnregisterDestroyedHeaderFooter( HeaderFooter* ) ) );
        headerFooter->setParent( 0 );
    }
    d->slotRelayout();
}
```

7.20.3.35 void Chart::takeLegend (Legend ∗ *legend*)

Removes the legend from the chart, without deleting it.

The chart no longer owns the legend, so it is the caller's responsibility to delete the legend.

See also:

addLegend, takeLegend

Definition at line 1149 of file KDChartChart.cpp.

References d.

Referenced by replaceLegend().

```
{
    const int idx = d->legends.indexOf( legend );
    if( idx != -1 ){
        d->legends.takeAt( idx );
        disconnect( legend, SIGNAL( destroyedLegend( Legend* ) ),
                    d, SLOT( slotUnregisterDestroyedLegend( Legend* ) ) );
        legend->setParent( 0 );
    }
    d->slotRelayout();
}
```

7.20.4 Member Data Documentation

7.20.4.1 Q_SIGNALS KDChart::Chart::__pad0__

Definition at line 396 of file KDChartChart.h.

The documentation for this class was generated from the following files:

- KDChartChart.h
- KDChartChart.cpp

7.21 KDChart::DataDimension Class Reference

`#include <KDChartAbstractCoordinatePlane.h>`

Collaboration diagram for KDChart::DataDimension:

7.21.1 Detailed Description

Helper class for one dimension of data, e.g. for the rows in a data model, or for the labels of an axis, or for the vertical lines in a grid.

isCalculated specifies whether this dimension's values are calculated or counted. (counted == "Item 1", "Item 2", "Item 3" ...)

sequence is the GranularitySequence, as specified at for the respective coordinate plane.

Step width is an optional parameter, to be omitted (or set to Zero, resp.) if the step width is unknown.

The default c'tor just gets you counted values from 1..10, using step width 1, used by the CartesianGrid, when showing an empty plane without any diagrams.

Definition at line 333 of file KDChartAbstractCoordinatePlane.h.

Public Member Functions

- DataDimension (qreal start_, qreal end_, bool isCalculated_, AbstractCoordinatePlane::AxesCalcMode calcMode_, KDChartEnums::GranularitySequence sequence_, qreal stepWidth_=0.0, qreal subStepWidth_=0.0)
- DataDimension ()
- qreal distance () const

  Returns the size of the distance, equivalent to the width() (or height(), resp.) of a QRectF.

- bool operator!= (const DataDimension &other) const
- bool operator== (const DataDimension &r) const

Public Attributes

- AbstractCoordinatePlane::AxesCalcMode calcMode
- qreal end
- bool isCalculated
- KDChartEnums::GranularitySequence sequence
- qreal start
- qreal stepWidth
- qreal subStepWidth

7.21.2 Constructor & Destructor Documentation

7.21.2.1 KDChart::DataDimension::DataDimension ()

Definition at line 335 of file KDChartAbstractCoordinatePlane.h.

References calcMode, sequence, stepWidth, and subStepWidth.

```
339         {}
340         DataDimension( qreal start_,
341                        qreal end_,
342                        bool isCalculated_,
343                        AbstractCoordinatePlane::AxesCalcMode calcMode_,
```

7.21.2.2 KDChart::DataDimension::DataDimension (qreal *start_*, qreal *end_*, bool *isCalculated_*, AbstractCoordinatePlane::AxesCalcMode *calcMode_*, KDChartEnums::GranularitySequence *sequence_*, qreal *stepWidth_* = 0.0, qreal *subStepWidth_* = 0.0)

Definition at line 344 of file KDChartAbstractCoordinatePlane.h.

References calcMode, end, isCalculated, sequence, start, stepWidth, and subStepWidth.

```
347             : start(        start_ )
348             , end(          end_ )
349             , isCalculated( isCalculated_ )
350             , calcMode(     calcMode_ )
351             , sequence(     sequence_ )
352             , stepWidth(    stepWidth_ )
353             , subStepWidth( subStepWidth_ )
354         {}
```

7.21.3 Member Function Documentation

7.21.3.1 qreal KDChart::DataDimension::distance () const

Returns the size of the distance, equivalent to the width() (or height(), resp.) of a QRectF.

Note that this value can be negative, e.g. indicating axis labels going in reversed direction.

Definition at line 366 of file KDChartAbstractCoordinatePlane.h.

References start.

Referenced by KDChart::CartesianCoordinatePlane::layoutDiagrams(), and KDChart::CartesianAxis::paintCtx().

```
368         {
369             return
```

7.21.3.2 bool KDChart::DataDimension::operator!= (const DataDimension & *other*) const

Definition at line 383 of file KDChartAbstractCoordinatePlane.h.

References calcMode, end, and isCalculated.

7.21.3.3 bool KDChart::DataDimension::operator== (const DataDimension & *r*) const

Definition at line 371 of file KDChartAbstractCoordinatePlane.h.

References calcMode, end, isCalculated, sequence, stepWidth, and subStepWidth.

```
380         { return !operator==( other ); }
381
```

7.21.4 Member Data Documentation

7.21.4.1 AbstractCoordinatePlane::AxesCalcMode KDChart::DataDimension::calcMode

Definition at line 389 of file KDChartAbstractCoordinatePlane.h.

Referenced by DataDimension(), operator!=(), operator==(), and KDChart::CartesianAxis::paintCtx().

7.21.4.2 qreal KDChart::DataDimension::end

Definition at line 387 of file KDChartAbstractCoordinatePlane.h.

Referenced by DataDimension(), KDChart::CartesianCoordinatePlane::layoutDiagrams(), operator!=(), operator==(), and KDChart::CartesianAxis::paintCtx().

7.21.4.3 bool KDChart::DataDimension::isCalculated

Definition at line 388 of file KDChartAbstractCoordinatePlane.h.

Referenced by DataDimension(), operator!=(), operator==(), and KDChart::CartesianAxis::paintCtx().

7.21.4.4 KDChartEnums::GranularitySequence KDChart::DataDimension::sequence

Definition at line 390 of file KDChartAbstractCoordinatePlane.h.

Referenced by DataDimension(), and operator==().

7.21.4.5 qreal KDChart::DataDimension::start

Definition at line 386 of file KDChartAbstractCoordinatePlane.h.

Referenced by DataDimension(), distance(), KDChart::CartesianCoordinatePlane::layoutDiagrams(), and KDChart::CartesianAxis::paintCtx().

7.21.4.6 qreal KDChart::DataDimension::stepWidth

Definition at line 391 of file KDChartAbstractCoordinatePlane.h.

Referenced by DataDimension(), operator==(), and KDChart::CartesianAxis::paintCtx().

7.21.4.7 qreal KDChart::DataDimension::subStepWidth

Definition at line 392 of file KDChartAbstractCoordinatePlane.h.

Referenced by DataDimension(), operator==(), and KDChart::CartesianAxis::paintCtx().

The documentation for this class was generated from the following file:

- KDChartAbstractCoordinatePlane.h

7.22 KDChart::DatasetProxyModel Class Reference

`#include <KDChartDatasetProxyModel.h>`

Inheritance diagram for KDChart::DatasetProxyModel:Collaboration diagram for KDChart::DatasetProxyModel:

7.22.1 Detailed Description

DatasetProxyModel takes a KDChart dataset configuration and translates it into a filtering proxy model.

The resulting model will only contain the part of the model that is selected by the dataset, and the according row and column header data.

Currently, this model is implemented for table models only. The way it would work with models representing a tree is to be decided.

The column selection is configured by passing a dataset description vector to the model. This vector (of integers) is supposed to have one value for each column of the original model. If the value at position x is -1, column x of the original model is not included in the dataset. If it is between 0 and (columnCount() -1), it is the column the source column is mapped to in the resulting model. Any other value is an error.

Definition at line 58 of file KDChartDatasetProxyModel.h.

Public Member Functions

- QVariant data (const QModelIndex &index, int role) const

  Overloaded from base class.

- DatasetProxyModel (QObject ∗parent=0)

  Create a DatasetProxyModel.

- QVariant headerData (int section, Qt::Orientation orientation, int role=Qt::DisplayRole) const

  Overloaded from base class.

- QModelIndex index (int row, int column, const QModelIndex &parent=QModelIndex()) const
- QModelIndex mapFromSource (const QModelIndex &sourceIndex) const

  Implements the mapping from the source to the proxy indexes.

- QModelIndex mapToSource (const QModelIndex &proxyIndex) const

  Implements the mapping from the proxy to the source indexes.

- QModelIndex parent (const QModelIndex &child) const
- void setDatasetColumnDescriptionVector (const DatasetDescriptionVector &columnConfig)

  Configure the dataset selection for the columns.

- void setDatasetDescriptionVectors (const DatasetDescriptionVector &rowConfig, const DatasetDescriptionVector &columnConfig)

  Convenience method to configure rows and columns in one step.

- void setDatasetRowDescriptionVector (const DatasetDescriptionVector &rowConfig)

  Configure the dataset selection for the rows.

- void setSourceModel (QAbstractItemModel ∗sourceModel)

  Overloaded from base class.

- void setSourceRootIndex (const QModelIndex &rootIdx)

  Set the root index of the table in the source model.

Public Attributes

- public Q_SLOTS: void resetDatasetDescriptions()

Protected Member Functions

- bool filterAcceptsColumn (int sourceColumn, const QModelIndex &) const

  Decide whether the column is accepted.

- bool filterAcceptsRow (int source_row, const QModelIndex &source_parent) const

  Decide whether the row is accepted.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 DatasetProxyModel::DatasetProxyModel (QObject ∗ *parent* = 0) `[explicit]`

Create a DatasetProxyModel.

Without further configuration, this model is invalid.

See also:

setDatasetDescriptionVector

Definition at line 35 of file KDChartDatasetProxyModel.cpp.

```
    : QSortFilterProxyModel ( parent )
{
}
```

7.22.3 Member Function Documentation

7.22.3.1 QVariant DatasetProxyModel::data (const QModelIndex & *index*, int *role*) const

Overloaded from base class.

Definition at line 208 of file KDChartDatasetProxyModel.cpp.

```
{
    return sourceModel()->data( mapToSource ( index ), role );
}
```

7.22.3.2 bool DatasetProxyModel::filterAcceptsColumn (int *sourceColumn*, const QModelIndex &) const `[protected]`

Decide whether the column is accepted.

Definition at line 135 of file KDChartDatasetProxyModel.cpp.

```
137 {
138     if ( mColSrcToProxyMap.isEmpty() )
139     {   // no column mapping set up yet, all columns are passed down:
140         return true;
141     } else {
142         Q_ASSERT ( sourceModel() );
143         Q_ASSERT ( mColSrcToProxyMap.size() == sourceModel()->columnCount(mRootIndex) );
144         if ( mColSrcToProxyMap[sourceColumn] == -1 )
145         {   // this column is explicitly not accepted:
146             return false;
147         } else {
148             Q_ASSERT ( mColSrcToProxyMap[sourceColumn] >= 0
149                        && mColSrcToProxyMap[sourceColumn] < mColSrcToProxyMap.size() );
150             return true;
151         }
152     }
153 }
```

7.22.3.3 bool DatasetProxyModel::filterAcceptsRow (int *source_row*, const QModelIndex & *source_parent*) const `[protected]`

Decide whether the row is accepted.

Definition at line 115 of file KDChartDatasetProxyModel.cpp.

```
117 {
118     if ( mRowSrcToProxyMap.isEmpty() )
119     {   // no row mapping set, all rows are passed down:
120         return true;
121     } else {
122         Q_ASSERT ( sourceModel() );
123         Q_ASSERT ( mRowSrcToProxyMap.size() == sourceModel()->rowCount(mRootIndex) );
124         if ( mRowSrcToProxyMap[sourceRow] == -1 )
125         {   // this row is explicitly not accepted:
126             return false;
127         } else {
128             Q_ASSERT ( mRowSrcToProxyMap[sourceRow] >= 0
129                        && mRowSrcToProxyMap[sourceRow] < mRowSrcToProxyMap.size() );
130             return true;
131         }
132     }
133 }
```

7.22.3.4 QVariant DatasetProxyModel::headerData (int *section*, Qt::Orientation *orientation*, int *role* = Qt::DisplayRole) const

Overloaded from base class.

Definition at line 213 of file KDChartDatasetProxyModel.cpp.

```
214 {
215     if ( orientation == Qt::Horizontal )
```

```
    {
        if ( mapProxyColumnToSource ( section ) == -1 )
        {
            return QVariant();
        } else {
            return sourceModel()->headerData ( mapProxyColumnToSource ( section ),
                                                       orientation,  role );
        }
    } else {
        if ( mapProxyRowToSource ( section ) == -1 )
        {
            return QVariant();
        } else {
            return sourceModel()->headerData ( mapProxyRowToSource ( section ),
                                                       orientation, role );
        }
    }
}
```

7.22.3.5 QModelIndex DatasetProxyModel::index (int *row*, int *column*, const QModelIndex & *parent* = QModelIndex()) const

Definition at line 68 of file KDChartDatasetProxyModel.cpp.

References mapFromSource().

```
{
    return mapFromSource( sourceModel()->index( mapProxyRowToSource(row),
                                                mapProxyColumnToSource(column),
                                                parent ) );
}
```

7.22.3.6 QModelIndex DatasetProxyModel::mapFromSource (const QModelIndex & *sourceIndex*) const

Implements the mapping from the source to the proxy indexes.

Definition at line 81 of file KDChartDatasetProxyModel.cpp.

Referenced by index(), and parent().

```
{
    Q_ASSERT_X ( sourceModel(), "DatasetProxyModel::mapFromSource", "A source "
                 "model must be set before the selection can be configured." );

    if ( !sourceIndex.isValid() ) return sourceIndex;

    if ( mRowSrcToProxyMap.isEmpty() && mColSrcToProxyMap.isEmpty() )
    {
        return createIndex ( sourceIndex.row(), sourceIndex.column(),
                             sourceIndex.internalPointer() );
    } else {
        int row = mapSourceRowToProxy ( sourceIndex.row() );
        int column = mapSourceColumnToProxy ( sourceIndex.column() );
        return createIndex ( row, column, sourceIndex.internalPointer() );
    }
}
```

7.22.3.7 QModelIndex DatasetProxyModel::mapToSource (const QModelIndex & *proxyIndex*) const

Implements the mapping from the proxy to the source indexes.

Definition at line 99 of file KDChartDatasetProxyModel.cpp.

```
{
    Q_ASSERT_X ( sourceModel(), "DatasetProxyModel::mapToSource", "A source "
                 "model must be set before the selection can be configured." );

    if ( !proxyIndex.isValid() ) return proxyIndex;
    if ( mRowSrcToProxyMap.isEmpty() && mColSrcToProxyMap.isEmpty() )
    {
        return sourceModel()->index( proxyIndex.row(),  proxyIndex.column(), mRootIndex );
    } else {
        int row = mapProxyRowToSource ( proxyIndex.row() );
        int column = mapProxyColumnToSource ( proxyIndex.column() );
        return sourceModel()->index( row, column, mRootIndex );
    }
}
```

7.22.3.8 QModelIndex DatasetProxyModel::parent (const QModelIndex & *child*) const

Definition at line 76 of file KDChartDatasetProxyModel.cpp.

References mapFromSource().

```
{
    return mapFromSource( sourceModel()->parent( child ) );
}
```

7.22.3.9 void DatasetProxyModel::setDatasetColumnDescriptionVector (const DatasetDescriptionVector & *columnConfig*)

Configure the dataset selection for the columns.

Every call to this method resets the previous dataset description.

Definition at line 50 of file KDChartDatasetProxyModel.cpp.

References KDChart::DatasetDescriptionVector.

Referenced by setDatasetDescriptionVectors().

```
{
    Q_ASSERT_X ( sourceModel(), "DatasetProxyModel::setDatasetColumnDescriptionVector",
                 "A source model must be set before the selection can be configured." );
    initializeDatasetDecriptors ( configuration, sourceModel()->columnCount(mRootIndex),
                                  mColSrcToProxyMap, mColProxyToSrcMap );
    clear(); // clear emits layoutChanged()
}
```

7.22.3.10 void DatasetProxyModel::setDatasetDescriptionVectors (const DatasetDescriptionVector & *rowConfig*, const DatasetDescriptionVector & *columnConfig*)

Convenience method to configure rows and columns in one step.

Definition at line 60 of file KDChartDatasetProxyModel.cpp.

References KDChart::DatasetDescriptionVector, setDatasetColumnDescriptionVector(), and setDatasetRowDescriptionVector().

```
{
    setDatasetRowDescriptionVector( rowConfig );
    setDatasetColumnDescriptionVector ( columnConfig );
}
```

7.22.3.11 void DatasetProxyModel::setDatasetRowDescriptionVector (const DatasetDescriptionVector & *rowConfig*)

Configure the dataset selection for the rows.

Every call to this method resets the previous dataset description.

Definition at line 40 of file KDChartDatasetProxyModel.cpp.

References KDChart::DatasetDescriptionVector.

Referenced by setDatasetDescriptionVectors().

```
{
    Q_ASSERT_X ( sourceModel(), "DatasetProxyModel::setDatasetRowDescriptionVector",
                 "A source model must be set before the selection can be configured." );
    initializeDatasetDecriptors ( configuration, sourceModel()->rowCount(mRootIndex),
                                  mRowSrcToProxyMap,  mRowProxyToSrcMap );
    clear(); // clear emits layoutChanged()
}
```

7.22.3.12 void DatasetProxyModel::setSourceModel (QAbstractItemModel ∗ *sourceModel*)

Overloaded from base class.

Definition at line 260 of file KDChartDatasetProxyModel.cpp.

```
{
    QSortFilterProxyModel::setSourceModel ( sourceModel );
    mRootIndex = QModelIndex();
    connect ( sourceModel,  SIGNAL ( layoutChanged() ),
              SLOT( resetDatasetDescriptions() ) );

    resetDatasetDescriptions();
}
```

7.22.3.13 void DatasetProxyModel::setSourceRootIndex (const QModelIndex & *rootIdx*)

Set the root index of the table in the source model.

Definition at line 270 of file KDChartDatasetProxyModel.cpp.

```
{
    mRootIndex = rootIdx;
    resetDatasetDescriptions();
}
```

7.22.4 Member Data Documentation

7.22.4.1 public KDChart::DatasetProxyModel::Q_SLOTS

Definition at line 97 of file KDChartDatasetProxyModel.h.

The documentation for this class was generated from the following files:

- KDChartDatasetProxyModel.h
- KDChartDatasetProxyModel.cpp

7.23 KDChart::DatasetSelectorWidget Class Reference

`#include <KDChartDatasetSelector.h>`

Inheritance diagram for KDChart::DatasetSelectorWidget:Collaboration diagram for KDChart::DatasetSelectorWidget:

Public Member Functions

- DatasetSelectorWidget (QWidget ∗parent=0)
- void mappingDisabled ()
- void on_cbReverseColumns_stateChanged (int)
- void on_cbReverseRows_stateChanged (int)
- void on_groupBox_toggled (bool)
- void on_sbColumnCount_valueChanged (int)
- void on_sbRowCount_valueChanged (int)
- void on_sbStartRow_valueChanged (int)
- void setSourceColumnCount (const int &columnCount)

Public Attributes

- Q_SIGNALS __pad0__: void configureDatasetProxyModel (const DatasetDescriptionVector& rowConfig
- Q_SIGNALS const DatasetDescriptionVector & columnConfig
- private Q_SLOTS: void on_sbStartColumn_valueChanged (int)
- public Q_SLOTS: void setSourceRowCount (const int& rowCount)

7.23.1 Constructor & Destructor Documentation

7.23.1.1 DatasetSelectorWidget::DatasetSelectorWidget (QWidget ∗ *parent* = 0) `[explicit]`

Definition at line 36 of file KDChartDatasetSelector.cpp.

```
    : QFrame ( parent )
    , mUi ( new Ui::DatasetSelector () )
    , mSourceRowCount ( 0 )
    , mSourceColumnCount ( 0 )
{
    qWarning("For DatasetSelectorWidget to become useful, it has to be connected to the proxy model it

    mUi->setupUi ( this );
    setMinimumSize ( minimumSizeHint() );
}
```

7.23.2 Member Function Documentation

7.23.2.1 void KDChart::DatasetSelectorWidget::mappingDisabled ()

Referenced by on_groupBox_toggled().

7.23.2.2 void DatasetSelectorWidget::on_cbReverseColumns_stateChanged (int)

Definition at line 73 of file KDChartDatasetSelector.cpp.

```
74 {
75     calculateMapping();
76 }
```

7.23.2.3 void DatasetSelectorWidget::on_cbReverseRows_stateChanged (int)

Definition at line 68 of file KDChartDatasetSelector.cpp.

```
69 {
70     calculateMapping();
71 }
```

7.23.2.4 void DatasetSelectorWidget::on_groupBox_toggled (bool)

Definition at line 78 of file KDChartDatasetSelector.cpp.

References mappingDisabled().

```
79 {
80     if ( state )
81     {
82         calculateMapping();
83     } else {
84         emit mappingDisabled();
85     }
86 }
```

7.23.2.5 void DatasetSelectorWidget::on_sbColumnCount_valueChanged (int)

Definition at line 58 of file KDChartDatasetSelector.cpp.

```
59 {
60     calculateMapping();
61 }
```

7.23.2.6 void DatasetSelectorWidget::on_sbRowCount_valueChanged (int)

Definition at line 63 of file KDChartDatasetSelector.cpp.

```
64 {
65     calculateMapping();
66 }
```

7.23.2.7 void DatasetSelectorWidget::on_sbStartRow_valueChanged (int)

Definition at line 53 of file KDChartDatasetSelector.cpp.

```
54 {
55     calculateMapping();
56 }
```

7.23.2.8 void DatasetSelectorWidget::setSourceColumnCount (const int & *columnCount*)

Definition at line 98 of file KDChartDatasetSelector.cpp.

```
99 {
100     if ( columnCount != mSourceColumnCount )
101     {
102         mSourceColumnCount = columnCount;
103         resetDisplayValues();
104     }
105 }
```

7.23.3 Member Data Documentation

7.23.3.1 Q_SIGNALS KDChart::DatasetSelectorWidget::__pad0__

Definition at line 61 of file KDChartDatasetSelector.h.

7.23.3.2 Q_SIGNALS const DatasetDescriptionVector& KDChart::DatasetSelectorWidget::columnConfig

Definition at line 61 of file KDChartDatasetSelector.h.

7.23.3.3 private KDChart::DatasetSelectorWidget::Q_SLOTS

Definition at line 67 of file KDChartDatasetSelector.h.

7.23.3.4 public KDChart::DatasetSelectorWidget::Q_SLOTS

Definition at line 56 of file KDChartDatasetSelector.h.

The documentation for this class was generated from the following files:

- KDChartDatasetSelector.h
- KDChartDatasetSelector.cpp

7.24 KDChart::DataValueAttributes Class Reference

`#include <KDChartDataValueAttributes>`

7.24.1 Detailed Description

Diagram attributes dealing with data value labels.

The DataValueAttributes group all properties that can be set wrt data value labels and if and how they are displayed. This includes things like the text attributes (font, color), what markers are used, howmany decimal digits are displayed, etc.

Definition at line 59 of file KDChartDataValueAttributes.h.

Public Member Functions

- BackgroundAttributes backgroundAttributes () const
- QString dataLabel () const

  Returns the string displayed instead of the data value label.

- DataValueAttributes (const DataValueAttributes &)
- DataValueAttributes ()
- int decimalDigits () const
- FrameAttributes frameAttributes () const
- bool isVisible () const
- MarkerAttributes markerAttributes () const
- const RelativePosition negativePosition () const

  Return the relative positioning of the data value labels.

- bool operator!= (const DataValueAttributes &other) const
- DataValueAttributes & operator= (const DataValueAttributes &)
- bool operator== (const DataValueAttributes &) const
- const RelativePosition position (bool positive) const
- const RelativePosition positivePosition () const

  Return the relative positioning of the data value labels.

- int powerOfTenDivisor () const
- QString prefix () const

  Returns the string used as a prefix to the data value text.

- void setBackgroundAttributes (const BackgroundAttributes &a)

  Set the background attributes to use for the data value labels area.

- void setDataLabel (const QString label)

  display a string label instead of the original data value label

- void setDecimalDigits (int digits)

  Set how many decimal digits to display when rendering the data value labels.

- void setFrameAttributes (const FrameAttributes &a)

Set the frame attributes to use for the data value labels area.

- void setMarkerAttributes (const MarkerAttributes &a)

  Set the marker attributes to use for the data values.

- void setNegativePosition (const RelativePosition &relPosition)

  Defines the relative positioning of the data value labels for negative values.

- void setPositivePosition (const RelativePosition &relPosition)

  Defines the relative position of the data value labels for positive values.

- void setPowerOfTenDivisor (int powerOfTenDivisor)

  These method are planned for future versions of KD Chart, so they are not part of the documented API yet.

- void setPrefix (const QString prefix)

  Prepend a prefix string to the data value label.

- void setShowInfinite (bool infinite)

  PLANNED_FOR_FUTURE

- void setShowRepetitiveDataLabels (bool showRepetitiveDataLabels)

  Set whether data value labels not different from their predecessors should be drawn.

- void setSuffix (const QString suffix)

  Append a suffix string to the data value label.

- void setTextAttributes (const TextAttributes &a)

  Set the text attributes to use for the data value labels.

- void setVisible (bool visible)

  Set whether data value labels should be displayed.

- bool showInfinite () const
- bool showRepetitiveDataLabels () const

  PLANNED_FOR_FUTURE

- QString suffix () const

  Returns the string used as a suffix to the data value text.

- TextAttributes textAttributes () const
- ∼DataValueAttributes ()

Static Public Member Functions

- const DataValueAttributes & defaultAttributes ()
- const QVariant & defaultAttributesAsVariant ()

7.24.2 Constructor & Destructor Documentation

7.24.2.1 KDChart::DataValueAttributes::DataValueAttributes ()

7.24.2.2 KDChart::DataValueAttributes::DataValueAttributes (const DataValueAttributes &)

7.24.2.3 KDChart::DataValueAttributes::∼DataValueAttributes ()

7.24.3 Member Function Documentation

7.24.3.1 BackgroundAttributes KDChart::DataValueAttributes::backgroundAttributes () const

Returns:

The background attributes used for painting the data value labels area.

See also:

BackgroundAttributes

Referenced by operator<<().

7.24.3.2 QString KDChart::DataValueAttributes::dataLabel () const

Returns the string displayed instead of the data value label.

See also:

setDataLabel

Referenced by KDChart::AbstractDiagram::paintDataValueText().

7.24.3.3 int KDChart::DataValueAttributes::decimalDigits () const

Returns:

The number of decimal digits displayed.

Referenced by operator<<(), and KDChart::AbstractDiagram::paintDataValueText().

7.24.3.4 const DataValueAttributes& KDChart::DataValueAttributes::defaultAttributes () `[static]`

7.24.3.5 const QVariant& KDChart::DataValueAttributes::defaultAttributesAsVariant () `[static]`

7.24.3.6 FrameAttributes KDChart::DataValueAttributes::frameAttributes () const

Returns:

The frame attributes used for painting the data value labels area.

See also:

FrameAttributes

Referenced by operator<<().

7.24.3.7 bool KDChart::DataValueAttributes::isVisible () const

Returns:

Whether data value labels should be displayed.

Referenced by operator<<(), KDChart::AbstractDiagram::paintDataValueText(), and KDChart::AbstractDiagram::paintMarker().

7.24.3.8 MarkerAttributes KDChart::DataValueAttributes::markerAttributes () const

Returns:

The marker attributes used for decorating the data values.

See also:

MarkerAttributes

Referenced by KDChart::AbstractDiagram::paintMarker().

7.24.3.9 const RelativePosition KDChart::DataValueAttributes::negativePosition () const

Return the relative positioning of the data value labels.

See also:

setNegativePosition

Referenced by operator<<().

7.24.3.10 bool KDChart::DataValueAttributes::operator!= (const DataValueAttributes & *other*) const

Definition at line 66 of file KDChartDataValueAttributes.h.

```
66 { return !operator==(other); }
```

7.24.3.11 DataValueAttributes& KDChart::DataValueAttributes::operator= (const DataValueAttributes &)

7.24.3.12 bool KDChart::DataValueAttributes::operator== (const DataValueAttributes &) const

7.24.3.13 const RelativePosition KDChart::DataValueAttributes::position (bool *positive*) const

Definition at line 259 of file KDChartDataValueAttributes.h.

Referenced by KDChart::AbstractDiagram::paintDataValueText().

```
260   {
261     return positive ? positivePosition() : negativePosition();
262   }
```

7.24.3.14 const RelativePosition KDChart::DataValueAttributes::positivePosition () const

Return the relative positioning of the data value labels.

See also:
setPositivePosition

Referenced by operator<<().

7.24.3.15 int KDChart::DataValueAttributes::powerOfTenDivisor () const

Referenced by operator<<().

7.24.3.16 QString KDChart::DataValueAttributes::prefix () const

Returns the string used as a prefix to the data value text.

See also:
setPrefix

Referenced by KDChart::AbstractDiagram::paintDataValueText().

7.24.3.17 void KDChart::DataValueAttributes::setBackgroundAttributes (const BackgroundAttributes & *a*)

Set the background attributes to use for the data value labels area.

Parameters:
a The background attributes to set.

See also:
BackgroundAttributes

7.24.3.18 void KDChart::DataValueAttributes::setDataLabel (const QString *label*)

display a string label instead of the original data value label

See also:
dataLabel

7.24.3.19 void KDChart::DataValueAttributes::setDecimalDigits (int *digits*)

Set how many decimal digits to display when rendering the data value labels.

If there are no decimal digits it will not be displayed.

Parameters:
digits The number of decimal digits to use.

7.24.3.20 void KDChart::DataValueAttributes::setFrameAttributes (const FrameAttributes & *a*)

Set the frame attributes to use for the data value labels area.

Parameters:

a The frame attributes to set.

See also:

FrameAttributes

7.24.3.21 void KDChart::DataValueAttributes::setMarkerAttributes (const MarkerAttributes & *a*)

Set the marker attributes to use for the data values.

This includes the marker type.

Parameters:

a The marker attributes to set.

See also:

MarkerAttributes

7.24.3.22 void KDChart::DataValueAttributes::setNegativePosition (const RelativePosition & *relPosition*)

Defines the relative positioning of the data value labels for negative values.

The position is specified in relation to the respective data value point, or in releation to the respective data representation area, that's one area segment in a LineDiagram showing areas, or one bar in a BarDiagram, one pie slice ...

See also:

negativePosition

7.24.3.23 void KDChart::DataValueAttributes::setPositivePosition (const RelativePosition & *relPosition*)

Defines the relative position of the data value labels for positive values.

The position is specified in relation to the respective data value point, or in releation to the respective data representation area, that's one area segment in a LineDiagram showing areas, or one bar in a BarDiagram, one pie slice ...

See also:

positivePosition

7.24.3.24 void KDChart::DataValueAttributes::setPowerOfTenDivisor (int *powerOfTenDivisor*)

These method are planned for future versions of KD Chart, so they are not part of the documented API yet.

7.24.3.25 void KDChart::DataValueAttributes::setPrefix (const QString *prefix*)

Prepend a prefix string to the data value label.

See also:
prefix

7.24.3.26 void KDChart::DataValueAttributes::setShowInfinite (bool *infinite*)

PLANNED_FOR_FUTURE

These method are planned for future versions of KD Chart, so they are not part of the documented API yet.

7.24.3.27 void KDChart::DataValueAttributes::setShowRepetitiveDataLabels (bool *showRepetitiveDataLabels*)

Set whether data value labels not different from their predecessors should be drawn.

Parameters:
showRepetitiveDataLabels Whether data value not different from their predecessors are drawn.

7.24.3.28 void KDChart::DataValueAttributes::setSuffix (const QString *suffix*)

Append a suffix string to the data value label.

See also:
suffix

7.24.3.29 void KDChart::DataValueAttributes::setTextAttributes (const TextAttributes & *a*)

Set the text attributes to use for the data value labels.

Parameters:
a The text attributes to set.

See also:
TextAttributes

7.24.3.30 void KDChart::DataValueAttributes::setVisible (bool *visible*)

Set whether data value labels should be displayed.

Parameters:
visible Whether data value labels should be displayed.

7.24.3.31 bool KDChart::DataValueAttributes::showInfinite () const

Referenced by operator<<().

7.24.3.32 bool KDChart::DataValueAttributes::showRepetitiveDataLabels () const

PLANNED_FOR_FUTURE

Returns:
Whether data values not different from their predecessors are drawn.

Referenced by operator<<(), and KDChart::AbstractDiagram::paintDataValueText().

7.24.3.33 QString KDChart::DataValueAttributes::suffix () const

Returns the string used as a suffix to the data value text.

See also:
setSuffix

Referenced by KDChart::AbstractDiagram::paintDataValueText().

7.24.3.34 TextAttributes KDChart::DataValueAttributes::textAttributes () const

Returns:
The text attributes used for painting data value labels.

Referenced by operator<<(), and KDChart::AbstractDiagram::paintDataValueText().

The documentation for this class was generated from the following file:

- KDChartDataValueAttributes.h

7.25 KDChart::DiagramObserver Class Reference

`#include <KDChartDiagramObserver.h>`

Inheritance diagram for KDChart::DiagramObserver:Collaboration diagram for KDChart::DiagramObserver:

7.25.1 Detailed Description

A DiagramObserver watches the associated diagram for changes and deletion and emits corresponsing signals.

Definition at line 44 of file KDChartDiagramObserver.h.

Public Member Functions

- AbstractDiagram ∗ diagram ()
- const AbstractDiagram ∗ diagram () const
- void diagramAttributesChanged (AbstractDiagram ∗diagram)

  This signal is emitted whenever the attributes of the diagram change.

- void diagramDataChanged (AbstractDiagram ∗diagram)

  This signal is emitted whenever the data of the diagram changes.

- void diagramDataHidden (AbstractDiagram ∗diagram)

  This signal is emitted whenever any of the data of the diagram was set (un)hidden.

- DiagramObserver (AbstractDiagram ∗diagram, QObject ∗parent=0)

  Constructs a new observer observing the given diagram.

- void slotAttributesChanged (QModelIndex, QModelIndex)
- void slotAttributesChanged ()
- void slotDataChanged ()
- void slotDataChanged (QModelIndex, QModelIndex)
- void slotDataHidden ()
- void slotHeaderDataChanged (Qt::Orientation, int, int)
- void slotModelsChanged ()
- ∼DiagramObserver ()

Public Attributes

- Q_SIGNALS __pad0__: void diagramDestroyed(AbstractDiagram∗ diagram)
- private Q_SLOTS: void slotDestroyed(QObject∗)

7.25.2 Constructor & Destructor Documentation

7.25.2.1 DiagramObserver::DiagramObserver (AbstractDiagram ∗ *diagram*, QObject ∗ *parent* = 0) `[explicit]`

Constructs a new observer observing the given diagram.

Definition at line 40 of file KDChartDiagramObserver.cpp.

References slotModelsChanged().

```
    : QObject( parent ), m_diagram( diagram )
{
    if ( m_diagram ) {
        connect( m_diagram, SIGNAL(destroyed(QObject*)), SLOT(slotDestroyed(QObject*)));
        connect( m_diagram, SIGNAL(modelsChanged()), SLOT(slotModelsChanged()));
    }
    init();
}
```

7.25.2.2 DiagramObserver::∼DiagramObserver ()

Definition at line 50 of file KDChartDiagramObserver.cpp.

```
{
}
```

7.25.3 Member Function Documentation

7.25.3.1 AbstractDiagram ∗ DiagramObserver::diagram ()

Definition at line 59 of file KDChartDiagramObserver.cpp.

```
{
    return m_diagram;
}
```

7.25.3.2 const AbstractDiagram ∗ DiagramObserver::diagram () const

Definition at line 54 of file KDChartDiagramObserver.cpp.

Referenced by KDChart::Legend::datasetCount().

```
{
    return m_diagram;
}
```

7.25.3.3 void KDChart::DiagramObserver::diagramAttributesChanged (AbstractDiagram ∗ *diagram*)

This signal is emitted whenever the attributes of the diagram change.

Referenced by slotAttributesChanged().

7.25.3.4 void KDChart::DiagramObserver::diagramDataChanged (AbstractDiagram ∗ *diagram*)

This signal is emitted whenever the data of the diagram changes.

Referenced by slotDataChanged(), and slotHeaderDataChanged().

7.25.3.5 void KDChart::DiagramObserver::diagramDataHidden (AbstractDiagram * *diagram*)

This signal is emitted whenever any of the data of the diagram was set (un)hidden.

Referenced by slotDataHidden().

7.25.3.6 void DiagramObserver::slotAttributesChanged (QModelIndex, QModelIndex)

Definition at line 133 of file KDChartDiagramObserver.cpp.

References slotAttributesChanged().

```
{
    slotAttributesChanged();
}
```

7.25.3.7 void DiagramObserver::slotAttributesChanged ()

Definition at line 138 of file KDChartDiagramObserver.cpp.

References diagramAttributesChanged().

Referenced by slotAttributesChanged(), and slotModelsChanged().

```
{
    //qDebug() << "DiagramObserver::slotAttributesChanged()";
    emit diagramAttributesChanged( m_diagram );
}
```

7.25.3.8 void DiagramObserver::slotDataChanged ()

Definition at line 121 of file KDChartDiagramObserver.cpp.

References diagramDataChanged().

Referenced by slotDataChanged(), and slotModelsChanged().

```
{
    //qDebug() << "DiagramObserver::slotDataChanged()";
    emit diagramDataChanged( m_diagram );
}
```

7.25.3.9 void DiagramObserver::slotDataChanged (QModelIndex, QModelIndex)

Definition at line 116 of file KDChartDiagramObserver.cpp.

References slotDataChanged().

```
{
    slotDataChanged();
}
```

7.25.3.10 void DiagramObserver::slotDataHidden ()

Definition at line 127 of file KDChartDiagramObserver.cpp.

References diagramDataHidden().

```
{
    //qDebug() << "DiagramObserver::slotDataHidden()";
    emit diagramDataHidden( m_diagram );
}
```

7.25.3.11 void DiagramObserver::slotHeaderDataChanged (Qt::Orientation, int, int)

Definition at line 110 of file KDChartDiagramObserver.cpp.

References diagramDataChanged().

```
{
    //qDebug() << "DiagramObserver::slotHeaderDataChanged()";
    emit diagramDataChanged( m_diagram );
}
```

7.25.3.12 void DiagramObserver::slotModelsChanged ()

Definition at line 103 of file KDChartDiagramObserver.cpp.

References slotAttributesChanged(), and slotDataChanged().

Referenced by DiagramObserver().

```
{
    init();
    slotDataChanged();
    slotAttributesChanged();
}
```

7.25.4 Member Data Documentation

7.25.4.1 Q_SIGNALS KDChart::DiagramObserver::__pad0__

Definition at line 60 of file KDChartDiagramObserver.h.

7.25.4.2 private KDChart::DiagramObserver::Q_SLOTS

Definition at line 69 of file KDChartDiagramObserver.h.

The documentation for this class was generated from the following files:

- KDChartDiagramObserver.h
- KDChartDiagramObserver.cpp

7.26 KDChart::FrameAttributes Class Reference

`#include <KDChartFrameAttributes.h>`

Public Member Functions

- FrameAttributes (const FrameAttributes &)
- FrameAttributes ()
- bool isVisible () const
- bool operator!= (const FrameAttributes &other) const
- FrameAttributes & operator= (const FrameAttributes &)
- bool operator== (const FrameAttributes &) const
- int padding () const
- QPen pen () const
- void setPadding (int padding)
- void setPen (const QPen &pen)
- void setVisible (bool visible)
- ∼FrameAttributes ()

7.26.1 Constructor & Destructor Documentation

7.26.1.1 KDChart::FrameAttributes::FrameAttributes ()

7.26.1.2 KDChart::FrameAttributes::FrameAttributes (const FrameAttributes &)

7.26.1.3 KDChart::FrameAttributes::∼FrameAttributes ()

7.26.2 Member Function Documentation

7.26.2.1 bool KDChart::FrameAttributes::isVisible () const

Referenced by operator<<(), KDChart::AbstractAreaBase::paintFrameAttributes(), and updateCommonBrush().

7.26.2.2 bool KDChart::FrameAttributes::operator!= (const FrameAttributes & *other*) const

Definition at line 59 of file KDChartFrameAttributes.h.

```
59 { return !operator==(other); }
```

7.26.2.3 FrameAttributes& KDChart::FrameAttributes::operator= (const FrameAttributes &)

7.26.2.4 bool KDChart::FrameAttributes::operator== (const FrameAttributes &) const

7.26.2.5 int KDChart::FrameAttributes::padding () const

Referenced by operator<<().

7.26.2.6 QPen KDChart::FrameAttributes::pen () const

Referenced by operator<<(), and KDChart::AbstractAreaBase::paintFrameAttributes().

7.26.2.7 void KDChart::FrameAttributes::setPadding (int *padding*)

Referenced by KDChart::Chart::Chart().

7.26.2.8 void KDChart::FrameAttributes::setPen (const QPen & *pen*)

Referenced by KDChart::Chart::Chart().

7.26.2.9 void KDChart::FrameAttributes::setVisible (bool *visible*)

Referenced by KDChart::Chart::Chart().

The documentation for this class was generated from the following file:

- KDChartFrameAttributes.h

7.27 KDChart::GlobalMeasureScaling Class Reference

`#include <KDChartMeasure.h>`

Collaboration diagram for KDChart::GlobalMeasureScaling:

7.27.1 Detailed Description

Auxiliary class used by the KDChart::Measure and KDChart::Chart class.

Normally there should be no need to call any of these methods yourself.

They are used by KDChart::Chart::paint(QPainter∗, const QRect&) to adjust all of the relative Measures according to the target rectangle's size.

Default factors are (1.0, 1.0)

Definition at line 148 of file KDChartMeasure.h.

Public Member Functions

- GlobalMeasureScaling ()
- virtual ~GlobalMeasureScaling ()

Static Public Member Functions

- const QPair< qreal, qreal > currentFactors ()

  Returns the currently active factors.

- GlobalMeasureScaling ∗ instance ()
- void resetFactors ()

  Reset factors to the values active before the previous call of setFactors.

- void setFactors (qreal factorX, qreal factorY)

  Set new factors to be used by all Measure objects from now on.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 KDChart::GlobalMeasureScaling::GlobalMeasureScaling ()

Definition at line 187 of file KDChartMeasure.cpp.

```
188 {
189     mFactors.push( qMakePair(1.0, 1.0) );
190 }
```

7.27.2.2 KDChart::GlobalMeasureScaling::~GlobalMeasureScaling () `[virtual]`

Definition at line 192 of file KDChartMeasure.cpp.

```
193 {
194     // this space left empty intentionally
195 }
```

7.27.3 Member Function Documentation

7.27.3.1 const QPair< qreal, qreal > KDChart::GlobalMeasureScaling::currentFactors () `[static]`

Returns the currently active factors.

Definition at line 215 of file KDChartMeasure.cpp.

References instance(), and mFactors.

```
216 {
217     return instance()->mFactors.top();
218 }
```

7.27.3.2 GlobalMeasureScaling * KDChart::GlobalMeasureScaling::instance () `[static]`

Definition at line 197 of file KDChartMeasure.cpp.

References instance().

Referenced by currentFactors(), instance(), resetFactors(), and setFactors().

```
198 {
199     static GlobalMeasureScaling instance;
200     return &instance;
201 }
```

7.27.3.3 void KDChart::GlobalMeasureScaling::resetFactors () `[static]`

Reset factors to the values active before the previous call of setFactors.

This works on a stack, so recursive calls works fine, like: setFactors, setFactors, unserFactors, unsetFactors

Definition at line 208 of file KDChartMeasure.cpp.

References instance(), and mFactors.

```
209 {
210     // never remove the initial (1.0. 1.0) setting
211     if( instance()->mFactors.count() > 1 )
212         instance()->mFactors.pop();
213 }
```

7.27.3.4 void KDChart::GlobalMeasureScaling::setFactors (qreal *factorX*, qreal *factorY*) `[static]`

Set new factors to be used by all Measure objects from now on.

Previous values will be stored.

Definition at line 203 of file KDChartMeasure.cpp.

References instance(), and mFactors.

```
{
    instance()->mFactors.push( qMakePair(factorX, factorY) );
}
```

The documentation for this class was generated from the following files:

- KDChartMeasure.h
- KDChartMeasure.cpp

7.28 KDChart::GridAttributes Class Reference

`#include <KDChartGridAttributes.h>`

Public Member Functions

- bool adjustLowerBoundToGrid () const
- bool adjustUpperBoundToGrid () const
- GridAttributes (const GridAttributes &)
- GridAttributes ()
- KDChartEnums::GranularitySequence gridGranularitySequence () const
- QPen gridPen () const
- qreal gridStepWidth () const
- qreal gridSubStepWidth () const
- bool isGridVisible () const
- bool isSubGridVisible () const
- bool operator!= (const GridAttributes &other) const
- GridAttributes & operator= (const GridAttributes &)
- bool operator== (const GridAttributes &) const
- void setAdjustBoundsToGrid (bool adjustLower, bool adjustUpper)

  By default visible bounds of the data area are adjusted to match a main grid line.

- void setGridGranularitySequence (KDChartEnums::GranularitySequence sequence)

  Specify which granularity sequence is to be used to find a matching grid granularity.

- void setGridPen (const QPen &pen)
- void setGridStepWidth (qreal stepWidth=0.0)
- void setGridSubStepWidth (qreal subStepWidth=0.0)
- void setGridVisible (bool visible)
- void setSubGridPen (const QPen &pen)
- void setSubGridVisible (bool visible)
- void setZeroLinePen (const QPen &pen)
- QPen subGridPen () const
- QPen zeroLinePen () const
- ~GridAttributes ()

7.28.1 Constructor & Destructor Documentation

7.28.1.1 KDChart::GridAttributes::GridAttributes ()

7.28.1.2 KDChart::GridAttributes::GridAttributes (const GridAttributes &)

7.28.1.3 KDChart::GridAttributes::∼GridAttributes ()

7.28.2 Member Function Documentation

7.28.2.1 bool KDChart::GridAttributes::adjustLowerBoundToGrid () const

7.28.2.2 bool KDChart::GridAttributes::adjustUpperBoundToGrid () const

7.28.2.3 KDChartEnums::GranularitySequence KDChart::GridAttributes::gridGranularitySequence () const

Referenced by KDChart::CartesianCoordinatePlane::getDataDimensionsList().

7.28.2.4 QPen KDChart::GridAttributes::gridPen () const

Referenced by operator<<().

7.28.2.5 qreal KDChart::GridAttributes::gridStepWidth () const

Referenced by KDChart::CartesianCoordinatePlane::getDataDimensionsList(), and operator<<().

7.28.2.6 qreal KDChart::GridAttributes::gridSubStepWidth () const

Referenced by KDChart::CartesianCoordinatePlane::getDataDimensionsList(), and operator<<().

7.28.2.7 bool KDChart::GridAttributes::isGridVisible () const

Referenced by operator<<().

7.28.2.8 bool KDChart::GridAttributes::isSubGridVisible () const

Referenced by operator<<().

7.28.2.9 bool KDChart::GridAttributes::operator!= (const GridAttributes & *other*) const

Definition at line 104 of file KDChartGridAttributes.h.

```
104 { return !operator==(other); }
```

7.28.2.10 GridAttributes& KDChart::GridAttributes::operator= (const GridAttributes &)

7.28.2.11 bool KDChart::GridAttributes::operator== (const GridAttributes &) const

7.28.2.12 void KDChart::GridAttributes::setAdjustBoundsToGrid (bool *adjustLower*, bool *adjustUpper*)

By default visible bounds of the data area are adjusted to match a main grid line.

If you set the respective adjust flag to false the bound will not start at a grid line's value but it will be the exact value of the data range set.

See also:

CartesianCoordinatePlane::setHorizontalRange
CartesianCoordinatePlane::setVerticalRange

7.28.2.13 void KDChart::GridAttributes::setGridGranularitySequence (KDChartEnums::GranularitySequence *sequence*)

Specify which granularity sequence is to be used to find a matching grid granularity.

See details explained at KDChartEnums::GranularitySequence.

You might also want to use setAdjustBoundsToGrid for fine-tuning the start/end value.

See also:

setAdjustBoundsToGrid, GranularitySequence

7.28.2.14 void KDChart::GridAttributes::setGridPen (const QPen & *pen*)

7.28.2.15 void KDChart::GridAttributes::setGridStepWidth (qreal *stepWidth* = 0.0)

7.28.2.16 void KDChart::GridAttributes::setGridSubStepWidth (qreal *subStepWidth* = 0.0)

7.28.2.17 void KDChart::GridAttributes::setGridVisible (bool *visible*)

7.28.2.18 void KDChart::GridAttributes::setSubGridPen (const QPen & *pen*)

7.28.2.19 void KDChart::GridAttributes::setSubGridVisible (bool *visible*)

7.28.2.20 void KDChart::GridAttributes::setZeroLinePen (const QPen & *pen*)

7.28.2.21 QPen KDChart::GridAttributes::subGridPen () const

Referenced by operator<<().

7.28.2.22 QPen KDChart::GridAttributes::zeroLinePen () const

Referenced by operator<<().

The documentation for this class was generated from the following file:

- KDChartGridAttributes.h

7.29 KDChart::HeaderFooter Class Reference

`#include <KDChartHeaderFooter.h>`

Inheritance diagram for KDChart::HeaderFooter:Collaboration diagram for KDChart::HeaderFooter:

Public Types

- enum HeaderFooterType {
  Header,
  Footer }

Public Member Functions

- void alignToReferencePoint (const RelativePosition &position)
- const QObject ∗ autoReferenceArea () const
- BackgroundAttributes backgroundAttributes () const
- virtual HeaderFooter ∗ clone () const
- bool compare (const AbstractAreaBase ∗other) const

  Returns true if both areas have the same settings.

- bool compare (const HeaderFooter &other) const
- virtual Qt::Orientations expandingDirections () const

  pure virtual in QLayoutItem

- FrameAttributes frameAttributes () const
- virtual QRect geometry () const

  pure virtual in QLayoutItem

- void getFrameLeadings (int &left, int &top, int &right, int &bottom) const
- HeaderFooter (Chart ∗parent=0)
- virtual bool intersects (const TextLayoutItem &other, const QPoint &myPos, const QPoint &otherPos) const
- virtual bool intersects (const TextLayoutItem &other, const QPointF &myPos, const QPointF &otherPos) const
- virtual bool isEmpty () const

  pure virtual in QLayoutItem

- virtual QSize maximumSize () const

  pure virtual in QLayoutItem

- virtual QSize minimumSize () const

  pure virtual in QLayoutItem

- virtual void paint (QPainter ∗)
- void paintAll (QPainter &painter)

  Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

- virtual void paintBackground (QPainter &painter, const QRect &rectangle)
- virtual void paintCtx (PaintContext ∗context)

  Default impl: Paint the complete item using its layouted position and size.

- virtual void paintFrame (QPainter &painter, const QRect &rectangle)
- virtual void paintIntoRect (QPainter &painter, const QRect &rect)

  Draws the background and frame, then calls paint().

- QLayout ∗ parentLayout ()
- Position position () const
- void positionChanged (HeaderFooter ∗)
- virtual QFont realFont () const
- virtual qreal realFontSize () const
- void removeFromParentLayout ()
- void setAutoReferenceArea (const QObject ∗area)
- void setBackgroundAttributes (const BackgroundAttributes &a)
- void setFrameAttributes (const FrameAttributes &a)
- virtual void setGeometry (const QRect &r)

  pure virtual in QLayoutItem

- void setParent (QObject ∗parent)
- void setParentLayout (QLayout ∗lay)
- virtual void setParentWidget (QWidget ∗widget)

  Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

- void setPosition (Position position)
- void setText (const QString &text)
- void setTextAttributes (const TextAttributes &a)

  Use this to specify the text attributes to be used for this item.

- void setType (HeaderFooterType type)
- virtual QSize sizeHint () const

  pure virtual in QLayoutItem

- virtual void sizeHintChanged () const

  Report changed size hint: ask the parent widget to recalculate the layout.

- QString text () const
- TextAttributes textAttributes () const

  Returns the text attributes to be used for this item.

- HeaderFooterType type () const
- virtual ∼HeaderFooter ()

Static Public Member Functions

- void paintBackgroundAttributes (QPainter &painter, const QRect &rectangle, const KDChart::BackgroundAttributes &attributes)
- void paintFrameAttributes (QPainter &painter, const QRect &rectangle, const KDChart::FrameAttributes &attributes)

Public Attributes

- Q_SIGNALS __pad0__: void destroyedHeaderFooter(HeaderFooter∗)

Protected Member Functions

- virtual QRect areaGeometry () const
- QRect innerRect () const
- virtual void positionHasChanged ()

Protected Attributes

- QWidget ∗ mParent
- QLayout ∗ mParentLayout

7.29.1 Member Enumeration Documentation

7.29.1.1 enum KDChart::HeaderFooter::HeaderFooterType

Enumeration values:

Header

Footer

Definition at line 56 of file KDChartHeaderFooter.h.

```
56                            { Header,
57                              Footer };
```

7.29.2 Constructor & Destructor Documentation

7.29.2.1 HeaderFooter::HeaderFooter (Chart ∗ *parent* = 0)

Definition at line 55 of file KDChartHeaderFooter.cpp.

References setParent().

Referenced by clone().

```
55                                        :
56      TextArea( new Private() )
57 {
58      setParent( parent );
59      init();
60 }
```

7.29.2.2 HeaderFooter::∼HeaderFooter () `[virtual]`

Definition at line 62 of file KDChartHeaderFooter.cpp.

```
63 {
64      emit destroyedHeaderFooter( this );
65 }
```

7.29.3 Member Function Documentation

7.29.3.1 void AbstractAreaBase::alignToReferencePoint (const RelativePosition & *position*) `[inherited]`

Definition at line 90 of file KDChartAbstractAreaBase.cpp.

```
{
    Q_UNUSED( position );
    // PENDING(kalle) FIXME
    qWarning( "Sorry, not implemented: void AbstractAreaBase::alignToReferencePoint( const RelativePosi
}
```

7.29.3.2 QRect TextArea::areaGeometry () const `[protected, virtual, inherited]`

Implements KDChart::AbstractAreaBase.

Definition at line 105 of file KDChartTextArea.cpp.

References KDChart::TextLayoutItem::geometry().

Referenced by KDChart::TextArea::paintAll().

```
{
    return geometry();
}
```

7.29.3.3 const QObject ∗ KDChart::TextLayoutItem::autoReferenceArea () const `[inherited]`

Definition at line 135 of file KDChartLayoutItems.cpp.

Referenced by setParent().

```
{
    return mAutoReferenceArea;
}
```

7.29.3.4 BackgroundAttributes AbstractAreaBase::backgroundAttributes () const `[inherited]`

Definition at line 112 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by updateCommonBrush().

```
{
    return d->backgroundAttributes;
}
```

7.29.3.5 HeaderFooter * HeaderFooter::clone () const `[virtual]`

Definition at line 91 of file KDChartHeaderFooter.cpp.

References d, HeaderFooter(), position(), setPosition(), KDChart::TextLayoutItem::setTextAttributes(), setType(), KDChart::TextLayoutItem::textAttributes(), and type().

```
{
    HeaderFooter* headerFooter = new HeaderFooter( new Private( *d ), 0 );
    headerFooter->setType( type() );
    headerFooter->setPosition( position() );
    headerFooter->setTextAttributes( textAttributes() );
    return headerFooter;
}
```

7.29.3.6 bool AbstractAreaBase::compare (const AbstractAreaBase * *other*) const `[inherited]`

Returns true if both areas have the same settings.

Definition at line 75 of file KDChartAbstractAreaBase.cpp.

```
{
    if( other == this ) return true;
    if( ! other ){
        //qDebug() << "CartesianAxis::compare() cannot compare to Null pointer";
        return false;
    }
    /*
    qDebug() << "AbstractAreaBase:" << (frameAttributes() == other->frameAttributes())
        << (backgroundAttributes() == other->backgroundAttributes()) << "\n";
    */
    return  (frameAttributes()      == other->frameAttributes()) &&
            (backgroundAttributes() == other->backgroundAttributes());
}
```

7.29.3.7 bool HeaderFooter::compare (const HeaderFooter & *other*) const

Definition at line 100 of file KDChartHeaderFooter.cpp.

```
{
    return  (type()            == other.type()) &&
            (position()        == other.position()) &&
            // also compare members inherited from the base class:
            (autoReferenceArea() == other.autoReferenceArea()) &&
            (text()               == other.text()) &&
            (textAttributes()     == other.textAttributes());
}
```

7.29.3.8 Qt::Orientations KDChart::TextLayoutItem::expandingDirections () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 175 of file KDChartLayoutItems.cpp.

```
176 {
177     return 0; // Grow neither vertically nor horizontally
178 }
```

7.29.3.9 FrameAttributes AbstractAreaBase::frameAttributes () const `[inherited]`

Definition at line 102 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone(), and updateCommonBrush().

```
103 {
104     return d->frameAttributes;
105 }
```

7.29.3.10 QRect KDChart::TextLayoutItem::geometry () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 180 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextArea::areaGeometry(), KDChart::TextLayoutItem::paint(), KDChart::TextArea::paintAll(), KDChart::CartesianAxis::paintCtx(), and KDChart::TextArea::paintIntoRect().

```
181 {
182     return mRect;
183 }
```

7.29.3.11 void AbstractAreaBase::getFrameLeadings (int & *left*, int & *top*, int & *right*, int & *bottom*) const `[inherited]`

Definition at line 204 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::AbstractAreaBase::innerRect(), and KDChart::AbstractAreaWidget::paintAll().

```
205 {
206     if( d && d->frameAttributes.isVisible() ){
207         const int padding = qMax( d->frameAttributes.padding(), 0 );
208         left   = padding;
209         top    = padding;
210         right  = padding;
211         bottom = padding;
212     }else{
213         left   = 0;
214         top    = 0;
215         right  = 0;
216         bottom = 0;
217     }
218 }
```

7.29.3.12 QRect AbstractAreaBase::innerRect () const `[protected, inherited]`

Definition at line 220 of file KDChartAbstractAreaBase.cpp.

References KDChart::AbstractAreaBase::areaGeometry(), and KDChart::AbstractAreaBase::getFrameLeadings().

Referenced by KDChart::TextArea::paintAll(), and KDChart::AbstractArea::paintAll().

```
221 {
222     int left;
223     int top;
224     int right;
225     int bottom;
226     getFrameLeadings( left, top, right, bottom );
227     return
228         QRect( QPoint(0,0), areaGeometry().size() )
229             .adjusted( left, top, -right, -bottom );
230 }
```

7.29.3.13 bool KDChart::TextLayoutItem::intersects (const TextLayoutItem & *other*, const QPoint & *myPos*, const QPoint & *otherPos*) const `[virtual, inherited]`

Definition at line 254 of file KDChartLayoutItems.cpp.

References KDChart::TextLayoutItem::mAttributes, PI, KDChart::TextLayoutItem::rotatedCorners(), KDChart::TextAttributes::rotation(), and KDChart::TextLayoutItem::unrotatedSizeHint().

```
255 {
256     if ( mAttributes.rotation() != other.mAttributes.rotation() )
257     {
258         // that’s the code for the common case: the rotation angles don’t need to match here
259         QPolygon myPolygon(          rotatedCorners() );
260         QPolygon otherPolygon( other.rotatedCorners() );
261
262         // move the polygons to their positions
263         myPolygon.translate( myPos );
264         otherPolygon.translate( otherPos );
265
266         // create regions out of it
267         QRegion myRegion( myPolygon );
268         QRegion otherRegion( otherPolygon );
269
270         // now the question - do they intersect or not?
271         return ! myRegion.intersect( otherRegion ).isEmpty();
272
273     } else {
274         // and that’s the code for the special case: the rotation angles match, which is less time cor
275         const qreal angle = mAttributes.rotation() * PI / 180.0;
276         // both sizes
277         const QSizeF mySize(          unrotatedSizeHint() );
278         const QSizeF otherSize( other.unrotatedSizeHint() );
279
280         // that’s myP1 relative to myPos
281         QPointF myP1( mySize.height() * sin( angle ), 0.0 );
282         // that’s otherP1 to myPos
283         QPointF otherP1 = QPointF( otherSize.height() * sin( angle ), 0.0 ) + otherPos - myPos;
284
285         // now rotate both points the negative angle around myPos
286         myP1 = QPointF( myP1.x() * cos( -angle ), myP1.x() * sin( -angle ) );
287         qreal r = sqrt( otherP1.x() * otherP1.x() + otherP1.y() * otherP1.y() );
288         otherP1 = QPointF( r * cos( -angle ), r * sin( -angle ) );
289
290         // finally we look, whether both rectangles intersect or even not
291         return QRectF( myP1, mySize ).intersects( QRectF( otherP1, otherSize ) );
292     }
293 }
```

7.29.3.14 bool KDChart::TextLayoutItem::intersects (const TextLayoutItem & *other*, const QPointF & *myPos*, const QPointF & *otherPos*) const `[virtual, inherited]`

Definition at line 249 of file KDChartLayoutItems.cpp.

Referenced by KDChart::CartesianAxis::paintCtx().

```
250 {
251     return intersects( other, myPos.toPoint(), otherPos.toPoint() );
252 }
```

7.29.3.15 bool KDChart::TextLayoutItem::isEmpty () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 185 of file KDChartLayoutItems.cpp.

```
186 {
187     return false; // never empty, otherwise the layout item would not exist
188 }
```

7.29.3.16 QSize KDChart::TextLayoutItem::maximumSize () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 190 of file KDChartLayoutItems.cpp.

References KDChart::TextLayoutItem::sizeHint().

```
191 {
192     return sizeHint(); // PENDING(kalle) Review, quite inflexible
193 }
```

7.29.3.17 QSize KDChart::TextLayoutItem::minimumSize () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 195 of file KDChartLayoutItems.cpp.

References KDChart::TextLayoutItem::sizeHint().

```
196 {
197     return sizeHint(); // PENDING(kalle) Review, quite inflexible
198 }
```

7.29.3.18 void KDChart::TextLayoutItem::paint (QPainter ∗) `[virtual, inherited]`

Implements KDChart::AbstractLayoutItem.

Definition at line 382 of file KDChartLayoutItems.cpp.

References KDChart::TextLayoutItem::geometry(), KDChart::TextAttributes::pen(), rotatedRect(), and KDChart::TextAttributes::rotation().

Referenced by KDChart::TextArea::paintAll(), and KDChart::CartesianAxis::paintCtx().

```
383 {
384     // make sure, cached font is updated, if needed:
385     // sizeHint();
386 
387     if( !mRect.isValid() )
388         return;
389 
390     PainterSaver painterSaver( painter );
391     painter->setFont( cachedFont );
392     QRectF rect( geometry() );
393 
394 // #ifdef DEBUG_ITEMS_PAINT
395 //     painter->setPen( Qt::black );
396 //     painter->drawRect( rect );
397 // #endif
398     painter->translate( rect.center() );
399     rect.moveTopLeft( QPointF( - rect.width() / 2, - rect.height() / 2 ) );
400 #ifdef DEBUG_ITEMS_PAINT
401     painter->setPen( Qt::blue );
402     painter->drawRect( rect );
403 #endif
404     painter->rotate( mAttributes.rotation() );
405     rect = rotatedRect( rect, mAttributes.rotation() );
406 #ifdef DEBUG_ITEMS_PAINT
407     painter->setPen( Qt::red );
408     painter->drawRect( rect );
409 #endif
410     painter->setPen( mAttributes.pen() );
411     painter->drawText( rect, Qt::AlignHCenter | Qt::AlignVCenter, mText );
412 //    if (  calcSizeHint( cachedFont ).width() > rect.width() )
413 //        qDebug() << "rect.width()" << rect.width() << "text.width()" << calcSizeHint( cachedFont ).w
414 //
415 //    //painter->drawText( rect, Qt::AlignHCenter | Qt::AlignVCenter, mText );
416 }
```

7.29.3.19 void TextArea::paintAll (QPainter & *painter*) `[virtual, inherited]`

Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

Reimplemented from KDChart::AbstractLayoutItem.

Definition at line 83 of file KDChartTextArea.cpp.

References KDChart::TextArea::areaGeometry(), KDChart::TextLayoutItem::geometry(), KDChart::AbstractAreaBase::innerRect(), KDChart::TextLayoutItem::paint(), KDChart::AbstractAreaBase::paintBackground(), KDChart::AbstractAreaBase::paintFrame(), and KDChart::TextLayoutItem::setGeometry().

Referenced by KDChart::TextArea::paintIntoRect().

```
84 {
    // Paint the background and frame
    paintBackground( painter, geometry() );
    paintFrame(      painter, geometry() );

    // temporarily adjust the widget size, to be sure all content gets calculated
    // to fit into the inner rectangle
    const QRect oldGeometry( areaGeometry()  );
    QRect inner( innerRect() );
    inner.moveTo(
        oldGeometry.left() + inner.left(),
        oldGeometry.top()  + inner.top() );
    const bool needAdjustGeometry = oldGeometry != inner;
    if( needAdjustGeometry )
        setGeometry( inner );
    paint( &painter );
    if( needAdjustGeometry )
        setGeometry( oldGeometry );
    //qDebug() << "TextAreaWidget::paintAll() done.";
}
```

7.29.3.20 void AbstractAreaBase::paintBackground (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 188 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintBackgroundAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintBackground()",
                "Private class was not initialized!" );
    paintBackgroundAttributes( painter, rect, d->backgroundAttributes );
}
```

7.29.3.21 void AbstractAreaBase::paintBackgroundAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::BackgroundAttributes & *attributes*) `[static, inherited]`

Definition at line 119 of file KDChartAbstractAreaBase.cpp.

References KDChart::BackgroundAttributes::brush(), KDChart::BackgroundAttributes::isVisible(), KDChart::BackgroundAttributes::pixmap(), and KDChart::BackgroundAttributes::pixmapMode().

Referenced by KDChart::AbstractAreaBase::paintBackground().

```
{
    if( !attributes.isVisible() ) return;

    /* first draw the brush (may contain a pixmap)*/
    if( Qt::NoBrush != attributes.brush().style() ) {
        KDChart::PainterSaver painterSaver( &painter );
        painter.setPen( Qt::NoPen );
        const QPointF newTopLeft( painter.deviceMatrix().map( rect.topLeft() ) );
        painter.setBrushOrigin( newTopLeft );
        painter.setBrush( attributes.brush() );
        painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
    }
    /* next draw the backPixmap over the brush */
    if( !attributes.pixmap().isNull() &&
        attributes.pixmapMode() != BackgroundAttributes::BackgroundPixmapModeNone ) {
        QPointF ol = rect.topLeft();
        if( BackgroundAttributes::BackgroundPixmapModeCentered == attributes.pixmapMode() )
        {
            ol.setX( rect.center().x() - attributes.pixmap().width() / 2 );
            ol.setY( rect.center().y() - attributes.pixmap().height()/ 2 );
            painter.drawPixmap( ol, attributes.pixmap() );
        } else {
            QMatrix m;
            double zW = (double)rect.width()  / (double)attributes.pixmap().width();
            double zH = (double)rect.height() / (double)attributes.pixmap().height();
            switch( attributes.pixmapMode() ) {
            case BackgroundAttributes::BackgroundPixmapModeScaled:
            {
                double z;
                z = qMin( zW, zH );
                m.scale( z, z );
            }
            break;
            case BackgroundAttributes::BackgroundPixmapModeStretched:
                m.scale( zW, zH );
                break;
            default:
                ; // Cannot happen, previously checked
            }
            QPixmap pm = attributes.pixmap().transformed( m );
            ol.setX( rect.center().x() - pm.width() / 2 );
            ol.setY( rect.center().y() - pm.height()/ 2 );
            painter.drawPixmap( ol, pm );
        }
    }
}
```

7.29.3.22 void KDChart::AbstractLayoutItem::paintCtx (PaintContext ∗ *context*) `[virtual, inherited]`

Default impl: Paint the complete item using its layouted position and size.

Reimplemented in KDChart::CartesianAxis.

Definition at line 77 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint(), and KDChart::PaintContext::painter().

```
{
    if( context )
        paint( context->painter() );
}
```

7.29.3.23 void AbstractAreaBase::paintFrame (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 196 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintFrameAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintFrame()",
                "Private class was not initialized!" );
    paintFrameAttributes( painter, rect, d->frameAttributes );
}
```

7.29.3.24 void AbstractAreaBase::paintFrameAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::FrameAttributes & *attributes*) `[static, inherited]`

Definition at line 169 of file KDChartAbstractAreaBase.cpp.

References KDChart::FrameAttributes::isVisible(), and KDChart::FrameAttributes::pen().

Referenced by KDChart::AbstractAreaBase::paintFrame().

```
{

    if( !attributes.isVisible() ) return;

    // Note: We set the brush to NoBrush explicitely here.
    //       Otherwise we might get a filled rectangle, so any
    //       previously drawn background would be overwritten by that area.

    const QPen   oldPen(   painter.pen() );
    const QBrush oldBrush( painter.brush() );
    painter.setPen(   attributes.pen() );
    painter.setBrush( Qt::NoBrush );
    painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
    painter.setBrush( oldBrush );
    painter.setPen(   oldPen );
}
```

7.29.3.25 void TextArea::paintIntoRect (QPainter & *painter*, const QRect & *rect*) `[virtual, inherited]`

Draws the background and frame, then calls paint().

In most cases there is no need to overwrite this method in a derived class, but you would overwrite TextLayoutItem::paint() instead.

Definition at line 71 of file KDChartTextArea.cpp.

References KDChart::TextLayoutItem::geometry(), KDChart::TextArea::paintAll(), and KDChart::TextLayoutItem::setGeometry().

```
{
    const QRect oldGeometry( geometry() );
    if( oldGeometry != rect )
        setGeometry( rect );
    painter.translate( rect.left(), rect.top() );
    paintAll( painter );
    painter.translate( -rect.left(), -rect.top() );
    if( oldGeometry != rect )
        setGeometry( oldGeometry );
}
```

7.29.3.26 QLayout∗ KDChart::AbstractLayoutItem::parentLayout () `[inherited]`

Definition at line 74 of file KDChartLayoutItems.h.

```
        {
            return mParentLayout;
        }
```

7.29.3.27 Position HeaderFooter::position () const

Definition at line 127 of file KDChartHeaderFooter.cpp.

References d.

Referenced by clone().

```
{
    return d->position;
}
```

7.29.3.28 void KDChart::HeaderFooter::positionChanged (HeaderFooter ∗)

Referenced by setPosition(), and setType().

7.29.3.29 void TextArea::positionHasChanged () `[protected, virtual, inherited]`

Reimplemented from KDChart::AbstractAreaBase.

Definition at line 110 of file KDChartTextArea.cpp.

```
{
    emit positionChanged( this );
}
```

7.29.3.30 QFont KDChart::TextLayoutItem::realFont () const `[virtual, inherited]`

Definition at line 226 of file KDChartLayoutItems.cpp.

Referenced by KDChart::CartesianAxis::maximumSize(), and KDChart::CartesianAxis::paintCtx().

```
{
    realFontWasRecalculated(); // we can safely ignore the boolean return value
    return cachedFont;
}
```

7.29.3.31 qreal KDChart::TextLayoutItem::realFontSize () const `[virtual, inherited]`

Definition at line 206 of file KDChartLayoutItems.cpp.

References KDChart::TextAttributes::calculatedFontSize().

```
{
    return mAttributes.calculatedFontSize( mAutoReferenceArea, mAutoReferenceOrientation );
}
```

7.29.3.32 void KDChart::AbstractLayoutItem::removeFromParentLayout () `[inherited]`

Definition at line 78 of file KDChartLayoutItems.h.

Referenced by KDChart::Chart::takeCoordinatePlane().

```
        {
            if( mParentLayout ){
                if( widget() )
                    mParentLayout->removeWidget( widget() );
                else
                    mParentLayout->removeItem( this );
            }
        }
```

7.29.3.33 void KDChart::TextLayoutItem::setAutoReferenceArea (const QObject ∗ *area*) `[inherited]`

Definition at line 128 of file KDChartLayoutItems.cpp.

References KDChart::TextLayoutItem::sizeHint().

Referenced by setParent().

```
{
    mAutoReferenceArea = area;
    cachedSizeHint = QSize();
    sizeHint();
}
```

7.29.3.34 void AbstractAreaBase::setBackgroundAttributes (const BackgroundAttributes & *a*) `[inherited]`

Definition at line 107 of file KDChartAbstractAreaBase.cpp.

References d.

```
{
    d->backgroundAttributes = a;
}
```

7.29.3.35 void AbstractAreaBase::setFrameAttributes (const FrameAttributes & *a*) `[inherited]`

Definition at line 97 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone().

```
{
    d->frameAttributes = a;
}
```

7.29.3.36 void KDChart::TextLayoutItem::setGeometry (const QRect & *r*) `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 200 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextArea::paintAll(), KDChart::CartesianAxis::paintCtx(), and KDChart::TextArea::paintIntoRect().

```
{
    mRect = r;
}
```

7.29.3.37 void HeaderFooter::setParent (QObject ∗ *parent*)

Definition at line 67 of file KDChartHeaderFooter.cpp.

References KDChart::TextLayoutItem::autoReferenceArea(), and KDChart::TextLayoutItem::setAutoReferenceArea().

Referenced by KDChart::Widget::addHeaderFooter(), KDChart::Chart::addHeaderFooter(), HeaderFooter(), KDChart::Widget::replaceHeaderFooter(), and KDChart::Chart::takeHeaderFooter().

```
{
    QObject::setParent( parent );
    if( parent && ! autoReferenceArea() )
        setAutoReferenceArea( parent );
}
```

7.29.3.38 void KDChart::AbstractLayoutItem::setParentLayout (QLayout ∗ *lay*) `[inherited]`

Definition at line 70 of file KDChartLayoutItems.h.

```
        {
            mParentLayout = lay;
        }
```

7.29.3.39 void KDChart::AbstractLayoutItem::setParentWidget (QWidget ∗ *widget*) `[virtual, inherited]`

Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

Thus, you need to call setParentWidget on every item, that has a non-fixed size.

Definition at line 64 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::mParent.

Referenced by KDChart::Legend::buildLegend(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
{
    mParent = widget;
}
```

7.29.3.40 void HeaderFooter::setPosition (Position *position*)

Definition at line 121 of file KDChartHeaderFooter.cpp.

References d, and positionChanged().

Referenced by KDChart::Widget::addHeaderFooter(), and clone().

```
{
    d->position = position;
    emit positionChanged( this );
}
```

7.29.3.41 void KDChart::TextLayoutItem::setText (const QString & *text*) `[inherited]`

Definition at line 140 of file KDChartLayoutItems.cpp.

References KDChart::TextLayoutItem::sizeHint().

Referenced by KDChart::Widget::addHeaderFooter(), KDChart::CartesianAxis::maximumSize(), and KDChart::CartesianAxis::paintCtx().

```
{
    mText = text;
    cachedSizeHint = QSize();
    sizeHint();
}
```

7.29.3.42 void KDChart::TextLayoutItem::setTextAttributes (const TextAttributes & *a*) `[inherited]`

Use this to specify the text attributes to be used for this item.

See also:
textAttributes

Definition at line 157 of file KDChartLayoutItems.cpp.

References KDChart::TextLayoutItem::sizeHint().

Referenced by clone().

```
{
    mAttributes = a;
    cachedSizeHint = QSize(); // invalidate size hint
    sizeHint();
}
```

7.29.3.43 void HeaderFooter::setType (HeaderFooterType *type*)

Definition at line 110 of file KDChartHeaderFooter.cpp.

References d, and positionChanged().

Referenced by KDChart::Widget::addHeaderFooter(), and clone().

```
111 {
112     d->type = type;
113     emit positionChanged( this );
114 }
```

7.29.3.44 QSize KDChart::TextLayoutItem::sizeHint () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 295 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::sizeHintChanged().

Referenced by KDChart::TextLayoutItem::maximumSize(), KDChart::CartesianAxis::maximumSize(), KDChart::TextLayoutItem::minimumSize(), KDChart::CartesianAxis::paintCtx(), KDChart::TextLayoutItem::setAutoReferenceArea(), KDChart::TextLayoutItem::setText(), and KDChart::TextLayoutItem::setTextAttributes().

```
296 {
297     if( realFontWasRecalculated() )
298     {
299         const QSize newSizeHint( calcSizeHint( cachedFont ) );
300         if( newSizeHint != cachedSizeHint ){
301             cachedSizeHint = newSizeHint;
302             sizeHintChanged();
303         }
304     }
305     //qDebug() << "-------- KDChart::TextLayoutItem::sizeHint() returns:"<<cachedSizeHint<<" ---------
306     return cachedSizeHint;
307 }
```

7.29.3.45 void KDChart::AbstractLayoutItem::sizeHintChanged () const `[virtual, inherited]`

Report changed size hint: ask the parent widget to recalculate the layout.

Definition at line 86 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextLayoutItem::sizeHint().

```
87 {
88     // This is exactly like what QWidget::updateGeometry does.
89 //  qDebug("KDChart::AbstractLayoutItem::sizeHintChanged() called");
90     if( mParent ) {
91         if ( mParent->layout() )
92             mParent->layout()->invalidate();
93         else
94             QApplication::postEvent( mParent, new QEvent( QEvent::LayoutRequest ) );
95     }
96 }
```

7.29.3.46 QString KDChart::TextLayoutItem::text () const `[inherited]`

Definition at line 147 of file KDChartLayoutItems.cpp.

Referenced by KDChart::CartesianAxis::paintCtx().

```
{
    return mText;
}
```

7.29.3.47 KDChart::TextAttributes KDChart::TextLayoutItem::textAttributes () const `[inherited]`

Returns the text attributes to be used for this item.

See also:

setTextAttributes

Definition at line 169 of file KDChartLayoutItems.cpp.

Referenced by clone().

```
{
    return mAttributes;
}
```

7.29.3.48 HeaderFooter::HeaderFooterType HeaderFooter::type () const

Definition at line 116 of file KDChartHeaderFooter.cpp.

References d.

Referenced by clone().

```
{
    return d->type;
}
```

7.29.4 Member Data Documentation

7.29.4.1 Q_SIGNALS KDChart::HeaderFooter::__pad0__

Reimplemented from KDChart::TextArea.

Definition at line 68 of file KDChartHeaderFooter.h.

7.29.4.2 QWidget∗ KDChart::AbstractLayoutItem::mParent `[protected, inherited]`

Definition at line 88 of file KDChartLayoutItems.h.

Referenced by KDChart::AbstractLayoutItem::setParentWidget().

7.29.4.3 QLayout∗ KDChart::AbstractLayoutItem::mParentLayout `[protected, inherited]`

Definition at line 89 of file KDChartLayoutItems.h.

The documentation for this class was generated from the following files:

- KDChartHeaderFooter.h
- KDChartHeaderFooter.cpp

7.30 KDChart::HorizontalLineLayoutItem Class Reference

`#include <KDChartLayoutItems.h>`

Inheritance diagram for KDChart::HorizontalLineLayoutItem:Collaboration diagram for KDChart::HorizontalLineLayoutItem:

Public Member Functions

- virtual Qt::Orientations expandingDirections () const
- virtual QRect geometry () const
- HorizontalLineLayoutItem ()
- virtual bool isEmpty () const
- virtual QSize maximumSize () const
- virtual QSize minimumSize () const
- virtual void paint (QPainter ∗)
- virtual void paintAll (QPainter &painter)

  Default impl: just call paint.

- virtual void paintCtx (PaintContext ∗context)

  Default impl: Paint the complete item using its layouted position and size.

- QLayout ∗ parentLayout ()
- void removeFromParentLayout ()
- virtual void setGeometry (const QRect &r)
- void setParentLayout (QLayout ∗lay)
- virtual void setParentWidget (QWidget ∗widget)

  Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

- virtual QSize sizeHint () const
- virtual void sizeHintChanged () const

  Report changed size hint: ask the parent widget to recalculate the layout.

Protected Attributes

- QWidget ∗ mParent
- QLayout ∗ mParentLayout

7.30.1 Constructor & Destructor Documentation

7.30.1.1 KDChart::HorizontalLineLayoutItem::HorizontalLineLayoutItem ()

Definition at line 418 of file KDChartLayoutItems.cpp.

```
419     : AbstractLayoutItem( Qt::AlignCenter )
420 {
421 }
```

7.30.2 Member Function Documentation

7.30.2.1 Qt::Orientations KDChart::HorizontalLineLayoutItem::expandingDirections () const `[virtual]`

Definition at line 423 of file KDChartLayoutItems.cpp.

```
{
    return Qt::Vertical|Qt::Horizontal; // Grow both vertically, and horizontally
}
```

7.30.2.2 QRect KDChart::HorizontalLineLayoutItem::geometry () const `[virtual]`

Definition at line 428 of file KDChartLayoutItems.cpp.

```
{
    return mRect;
}
```

7.30.2.3 bool KDChart::HorizontalLineLayoutItem::isEmpty () const `[virtual]`

Definition at line 433 of file KDChartLayoutItems.cpp.

```
{
    return false; // never empty, otherwise the layout item would not exist
}
```

7.30.2.4 QSize KDChart::HorizontalLineLayoutItem::maximumSize () const `[virtual]`

Definition at line 438 of file KDChartLayoutItems.cpp.

```
{
    return QSize( QWIDGETSIZE_MAX, QWIDGETSIZE_MAX );
}
```

7.30.2.5 QSize KDChart::HorizontalLineLayoutItem::minimumSize () const `[virtual]`

Definition at line 443 of file KDChartLayoutItems.cpp.

```
{
    return QSize( 0, 0 );
}
```

7.30.2.6 void KDChart::HorizontalLineLayoutItem::paint (QPainter ∗) `[virtual]`

Implements KDChart::AbstractLayoutItem.

Definition at line 459 of file KDChartLayoutItems.cpp.

```
460 {
461     if( !mRect.isValid() )
462         return;
463 
464     painter->drawLine( QPointF( mRect.left(), mRect.center().y() ),
465                        QPointF( mRect.right(), mRect.center().y() ) );
466 }
```

7.30.2.7 void KDChart::AbstractLayoutItem::paintAll (QPainter & *painter*) `[virtual, inherited]`

Default impl: just call paint.

Derived classes like KDChart::AbstractArea are providing additional action here.

Reimplemented in KDChart::AbstractArea, and KDChart::TextArea.

Definition at line 69 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint().

```
70 {
71     paint( &painter );
72 }
```

7.30.2.8 void KDChart::AbstractLayoutItem::paintCtx (PaintContext ∗ *context*) `[virtual, inherited]`

Default impl: Paint the complete item using its layouted position and size.

Reimplemented in KDChart::CartesianAxis.

Definition at line 77 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint(), and KDChart::PaintContext::painter().

```
78 {
79     if( context )
80         paint( context->painter() );
81 }
```

7.30.2.9 QLayout∗ KDChart::AbstractLayoutItem::parentLayout () `[inherited]`

Definition at line 74 of file KDChartLayoutItems.h.

```
75          {
76              return mParentLayout;
77          }
```

7.30.2.10 void KDChart::AbstractLayoutItem::removeFromParentLayout () `[inherited]`

Definition at line 78 of file KDChartLayoutItems.h.

Referenced by KDChart::Chart::takeCoordinatePlane().

```
        {
            if( mParentLayout ){
                if( widget() )
                    mParentLayout->removeWidget( widget() );
                else
                    mParentLayout->removeItem( this );
            }
        }
```

7.30.2.11 void KDChart::HorizontalLineLayoutItem::setGeometry (const QRect & *r*) `[virtual]`

Definition at line 448 of file KDChartLayoutItems.cpp.

```
{
    mRect = r;
}
```

7.30.2.12 void KDChart::AbstractLayoutItem::setParentLayout (QLayout ∗ *lay*) `[inherited]`

Definition at line 70 of file KDChartLayoutItems.h.

```
        {
            mParentLayout = lay;
        }
```

7.30.2.13 void KDChart::AbstractLayoutItem::setParentWidget (QWidget ∗ *widget*) `[virtual, inherited]`

Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

Thus, you need to call setParentWidget on every item, that has a non-fixed size.

Definition at line 64 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::mParent.

Referenced by KDChart::Legend::buildLegend(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
{
    mParent = widget;
}
```

7.30.2.14 QSize KDChart::HorizontalLineLayoutItem::sizeHint () const `[virtual]`

Definition at line 453 of file KDChartLayoutItems.cpp.

```
{
    return QSize( -1, 3 ); // see qframe.cpp
}
```

7.30.2.15 void KDChart::AbstractLayoutItem::sizeHintChanged () const `[virtual, inherited]`

Report changed size hint: ask the parent widget to recalculate the layout.

Definition at line 86 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextLayoutItem::sizeHint().

```
{
    // This is exactly like what QWidget::updateGeometry does.
//  qDebug("KDChart::AbstractLayoutItem::sizeHintChanged() called");
    if( mParent ) {
        if ( mParent->layout() )
            mParent->layout()->invalidate();
        else
            QApplication::postEvent( mParent, new QEvent( QEvent::LayoutRequest ) );
    }
}
```

7.30.3 Member Data Documentation

7.30.3.1 QWidget* KDChart::AbstractLayoutItem::mParent `[protected, inherited]`

Definition at line 88 of file KDChartLayoutItems.h.

Referenced by KDChart::AbstractLayoutItem::setParentWidget().

7.30.3.2 QLayout* KDChart::AbstractLayoutItem::mParentLayout `[protected, inherited]`

Definition at line 89 of file KDChartLayoutItems.h.

The documentation for this class was generated from the following files:

- KDChartLayoutItems.h
- KDChartLayoutItems.cpp

7.31 KDChartEnums Class Reference

`#include <KDChartEnums.h>`

Inheritance diagram for KDChartEnums:Collaboration diagram for KDChartEnums:

7.31.1 Detailed Description

Project global class providing some enums needed both by KDChartParams and by KDChartCustomBox.

Definition at line 46 of file KDChartEnums.h.

Public Types

- enum GranularitySequence {
  GranularitySequence_10_20,
  GranularitySequence_10_50,
  GranularitySequence_25_50,
  GranularitySequence_125_25,
  GranularitySequenceIrregular }

  GranularitySequence specifies the values, that may be applied, to determine a step width within a given data range.

- enum MeasureCalculationMode {
  MeasureCalculationModeAbsolute,
  MeasureCalculationModeRelative,
  MeasureCalculationModeAuto,
  MeasureCalculationModeAutoArea,
  MeasureCalculationModeAutoOrientation }

  Measure calculation mode: the way how the absolute value of a KDChart::Measure is determined during KD Chart's internal geometry calculation time.

- enum MeasureOrientation {
  MeasureOrientationAuto,
  MeasureOrientationHorizontal,
  MeasureOrientationVertical,
  MeasureOrientationMinimum,
  MeasureOrientationMaximum }

  Measure orientation mode: the way how the absolute value of a KDChart::Measure is determined during KD Chart's internal geometry calculation time.

- enum PositionValue {
  PositionUnknown = 0,
  PositionCenter = 1,
  PositionNorthWest = 2,
  PositionNorth = 3,

PositionNorthEast = 4,

PositionEast = 5,

PositionSouthEast = 6,

PositionSouth = 7,

PositionSouthWest = 8,

PositionWest = 9,

PositionFloating = 10 }

Numerical values of the static KDChart::Position instances, for using a Position::value() with a switch() statement.

- enum TextLayoutPolicy {

  LayoutJustOverwrite,

  LayoutPolicyRotate,

  LayoutPolicyShiftVertically,

  LayoutPolicyShiftHorizontally,

  LayoutPolicyShrinkFontSize }

  Text layout policy: what to do if text that is to be drawn would cover neighboring text or neighboring areas.

Static Public Member Functions

- QString granularitySequenceToString (GranularitySequence sequence)

  Converts the specified granularity sequence enum to a string representation.

- QString layoutPolicyToString (TextLayoutPolicy type)

  Converts the specified text layout policy enum to a string representation.

- QString measureCalculationModeToString (MeasureCalculationMode mode)

  Converts the specified measure calculation mode enum to a string representation.

- QString measureOrientationToString (MeasureOrientation mode)

  Converts the specified measure orientation enum to a string representation.

- GranularitySequence stringToGranularitySequence (const QString &string)

  Converts the specified string to a granularity sequence enum value.

- TextLayoutPolicy stringToLayoutPolicy (const QString &string)

  Converts the specified string to a text layout policy enum value.

- MeasureCalculationMode stringToMeasureCalculationMode (const QString &string)

  Converts the specified string to a measure calculation mode enum value.

- MeasureOrientation stringToMeasureOrientation (const QString &string)

  Converts the specified string to a measure orientation enum value.

7.31.2 Member Enumeration Documentation

7.31.2.1 enum KDChartEnums::GranularitySequence

GranularitySequence specifies the values, that may be applied, to determine a step width within a given data range.

Note:

Granularity with can be set for Linear axis calculation mode only, there is no way to specify a step width for Logarithmic axes.

Value occuring in the GranularitySequence names only are showing their respective relation ship. For real data they will most times not be used directly, but be multiplied by positive (or negative, resp.) powers of ten.

A granularity sequence is a sequence of values from the following set: 1, 1.25, 2, 2.5, 5.

The reason for using one of the following three pre-defined granularity sequences (instead of just using the best matching step width) is to follow a simple rule: If scaling becomes finer (== smaller step width) no value, that has been on a grid line before, shall loose its line and be NOT on a grid line anymore!

This means: Smaller step width may not remove any grid lines, but it may add additional lines in between.

- `GranularitySequence_10_20` Step widths can be 1, or 2, but they never can be 2.5 nor 5, nor 1.25.
- `GranularitySequence_10_50` Step widths can be 1, or 5, but they never can be 2, nor 2.5, nor 1.25.
- `GranularitySequence_25_50` Step widths can be 2.5, or 5, but they never can be 1, nor 2, nor 1.25.
- `GranularitySequence_125_25` Step widths can be 1.25 or 2.5 but they never can be 1, nor 2, nor 5.
- `GranularitySequenceIrregular` Step widths can be all of these values: 1, or 1.25, or 2, or 2.5, or 5.

Note:

When ever possible, try to avoid using GranularitySequenceIrregular! Allowing all possible step values, using this granularity sequence involves a serious risk: Your users might be irritated due to 'jumping' grid lines, when step size is changed from 2.5 to 2 (or vice versa, resp.). In case you still want to use GranularitySequenceIrregular just make sure to NOT draw any sub-grid lines, because in most cases you will get not-matching step widths for the sub-grid. In short: GranularitySequenceIrregular can safely be used if your data range is not changing at all AND (b) you will not allow the coordinate plane to be zoomed AND (c) you are not displaying any sub-grid lines.

Since you probably like having the value 1 as an allowed step width, the granularity sequence decision boils down to a boolean question:

- To get ten divided by five you use GranularitySequence_10_20, while
- for having it divided by two GranularitySequence_10_50 is your choice.

Enumeration values:

GranularitySequence_10_20

GranularitySequence_10_50

GranularitySequence_25_50

GranularitySequence_125_25

GranularitySequenceIrregular

Definition at line 101 of file KDChartEnums.h.

```
101                              {
102         GranularitySequence_10_20,
103         GranularitySequence_10_50,
104         GranularitySequence_25_50,
105         GranularitySequence_125_25,
106         GranularitySequenceIrregular };
```

7.31.2.2 enum KDChartEnums::MeasureCalculationMode

Measure calculation mode: the way how the absolute value of a KDChart::Measure is determined during KD Chart's internal geometry calculation time.

KDChart::Measure values either are relative (calculated in relation to a given AbstractArea), or they are absolute (used as fixed values).

Values stored in relative measure always are interpreted as per-mille of a reference area's height (or width, resp.) depending on the orientation set for the KDChart::Measure.

- `MeasureCalculationModeAbsolute` Value set by setValue() is absolute, to be used unchanged.
- `MeasureCalculationModeRelative` Value is relative, the reference area is specified by setReferenceArea(), and orientation specified by setOrientation().
- `MeasureCalculationModeAuto` Value is relative, KD Chart will automatically determine which reference area to use, and it will determine the orientation too.
- `MeasureCalculationModeAutoArea` Value is relative, Orientation is specified by setOrientation(), and KD Chart will automatically determine which reference area to use.
- `MeasureCalculationModeAutoOrientation` Value is relative, Area is specified by setReferenceArea(), and KD Chart will automatically determine which orientation to use.

See also:

KDChart::Measure::setCalculationMode

Enumeration values:

MeasureCalculationModeAbsolute

MeasureCalculationModeRelative

MeasureCalculationModeAuto

MeasureCalculationModeAutoArea

MeasureCalculationModeAutoOrientation

Definition at line 229 of file KDChartEnums.h.

```
229                                 { MeasureCalculationModeAbsolute,
230         MeasureCalculationModeRelative,
231         MeasureCalculationModeAuto,
232         MeasureCalculationModeAutoArea,
233         MeasureCalculationModeAutoOrientation };
```

7.31.2.3 enum KDChartEnums::MeasureOrientation

Measure orientation mode: the way how the absolute value of a KDChart::Measure is determined during KD Chart's internal geometry calculation time.

KDChart::Measure values either are relative (calculated in relation to a given AbstractArea), or they are absolute (used as fixed values).

Values stored in relative measure take into account the width (and/or the height, resp.) of a so-called reference area, that is either specified by KDChart::Measure::setReferenceArea, or determined by KD Chart automatically, respectively.

- `MeasureOrientationAuto` Value is calculated, based upon the width (or on the height, resp.) of the reference area: KD Chart will automatically determie an appropriate way.
- `MeasureOrientationHorizontal` Value is calculated, based upon the width of the reference area.
- `MeasureOrientationVertical` Value is calculated, based upon the height of the reference area.
- `MeasureOrientationMinimum` Value is calculated, based upon the width (or on the height, resp.) of the reference area - which ever is smaller.
- `MeasureOrientationMaximum` Value is calculated, based upon the width (or on the height, resp.) of the reference area - which ever is smaller.

See also:

KDChart::Measure::setOrientationMode

Enumeration values:

MeasureOrientationAuto

MeasureOrientationHorizontal

MeasureOrientationVertical

MeasureOrientationMinimum

MeasureOrientationMaximum

Definition at line 298 of file KDChartEnums.h.

```
298                             { MeasureOrientationAuto,
299         MeasureOrientationHorizontal,
300         MeasureOrientationVertical,
301         MeasureOrientationMinimum,
302         MeasureOrientationMaximum };
```

7.31.2.4 enum KDChartEnums::PositionValue

Numerical values of the static KDChart::Position instances, for using a Position::value() with a switch() statement.

See also:

Position

Enumeration values:

PositionUnknown

PositionCenter

PositionNorthWest

PositionNorth

PositionNorthEast

PositionEast

PositionSouthEast

PositionSouth

PositionSouthWest

PositionWest

PositionFloating

Definition at line 199 of file KDChartEnums.h.

```
199                         {
200          PositionUnknown   = 0,
201          PositionCenter    = 1,
202          PositionNorthWest = 2,
203          PositionNorth     = 3,
204          PositionNorthEast = 4,
205          PositionEast      = 5,
206          PositionSouthEast = 6,
207          PositionSouth     = 7,
208          PositionSouthWest = 8,
209          PositionWest      = 9,
210          PositionFloating  =10
211      };
```

7.31.2.5 enum KDChartEnums::TextLayoutPolicy

Text layout policy: what to do if text that is to be drawn would cover neighboring text or neighboring areas.

- `LayoutJustOverwrite` Just ignore the layout collision and write the text nevertheless.
- `LayoutPolicyRotate` Try counter-clockwise rotation to make the text fit into the space.
- `LayoutPolicyShiftVertically` Shift the text baseline upwards (or downwards, resp.) and draw a connector line between the text and its anchor.
- `LayoutPolicyShiftHorizontally` Shift the text baseline to the left (or to the right, resp.) and draw a connector line between the text and its anchor.
- `LayoutPolicyShrinkFontSize` Reduce the text font size.

See also:
KDChartParams::setPrintDataValues

Enumeration values:

LayoutJustOverwrite

LayoutPolicyRotate

LayoutPolicyShiftVertically

LayoutPolicyShiftHorizontally

LayoutPolicyShrinkFontSize

Definition at line 168 of file KDChartEnums.h.

```
168                           { LayoutJustOverwrite,
169         LayoutPolicyRotate,
170         LayoutPolicyShiftVertically,
171         LayoutPolicyShiftHorizontally,
172         LayoutPolicyShrinkFontSize };
```

7.31.3 Member Function Documentation

7.31.3.1 QString KDChartEnums::granularitySequenceToString (GranularitySequence *sequence*) `[static]`

Converts the specified granularity sequence enum to a string representation.

Parameters:

type the granularity sequence enum to convert

Returns:

the string representation of the granularity sequence

Definition at line 115 of file KDChartEnums.h.

```
115                                                                                {
116         switch( sequence ) {
117             case GranularitySequence_10_20:
118                 return QString::fromLatin1("GranularitySequence_10_20");
119             case GranularitySequence_10_50:
120                 return QString::fromLatin1("GranularitySequence_10_50");
121             case GranularitySequence_25_50:
122                 return QString::fromLatin1("GranularitySequence_25_50");
123             case GranularitySequence_125_25:
124                 return QString::fromLatin1("GranularitySequence_125_25");
125             case GranularitySequenceIrregular:
126                 return QString::fromLatin1("GranularitySequenceIrregular");
127             default: // should not happen
128         qDebug( "Unknown granularity sequence" );
129         return QString::fromLatin1("GranularitySequence_10_20");
130         }
131     }
```

7.31.3.2 QString KDChartEnums::layoutPolicyToString (TextLayoutPolicy *type*) `[static]`

Converts the specified text layout policy enum to a string representation.

Parameters:

type the text layout policy to convert

Returns:

the string representation of the text layout policy enum

7.31.3.3 QString KDChartEnums::measureCalculationModeToString (MeasureCalculationMode *mode*) `[static]`

Converts the specified measure calculation mode enum to a string representation.

Parameters:

type the measure calculation mode to convert

Returns:

the string representation of the Measure calculation mode enum

Definition at line 242 of file KDChartEnums.h.

```
242                                                                                  {
243         switch( mode ) {
244             case MeasureCalculationModeAbsolute:
245                 return QString::fromLatin1("MeasureCalculationModeAbsolute");
246             case MeasureCalculationModeAuto:
247                 return QString::fromLatin1("MeasureCalculationModeAuto");
248             case MeasureCalculationModeAutoArea:
249                 return QString::fromLatin1("MeasureCalculationModeAutoArea");
250             case MeasureCalculationModeAutoOrientation:
251                 return QString::fromLatin1("MeasureCalculationModeAutoOrientation");
252             case MeasureCalculationModeRelative:
253                 return QString::fromLatin1("MeasureCalculationModeRelative");
254             default: // should not happen
255         qDebug( "Unknown measure calculation mode" );
256         return QString::fromLatin1("MeasureCalculationModeAuto");
257         }
258     }
```

7.31.3.4 QString KDChartEnums::measureOrientationToString (MeasureOrientation *mode*) `[static]`

Converts the specified measure orientation enum to a string representation.

Parameters:

type the measure orientation to convert

Returns:

the string representation of the measure orientation enum

Definition at line 311 of file KDChartEnums.h.

```
311                                                                          {
312         switch( mode ) {
313             case MeasureOrientationAuto:
314                 return QString::fromLatin1("MeasureOrientationAuto");
315             case MeasureOrientationHorizontal:
316                 return QString::fromLatin1("MeasureOrientationHorizontal");
317             case MeasureOrientationVertical:
318                 return QString::fromLatin1("MeasureOrientationVertical");
319             case MeasureOrientationMinimum:
320                 return QString::fromLatin1("MeasureOrientationMinimum");
321             case MeasureOrientationMaximum:
322                 return QString::fromLatin1("MeasureOrientationMaximum");
323             default: // should not happen
324         qDebug( "Unknown measure orientation mode" );
325         return QString::fromLatin1("MeasureOrientationAuto");
326         }
327     }
```

7.31.3.5 GranularitySequence KDChartEnums::stringToGranularitySequence (const QString & *string*) `[static]`

Converts the specified string to a granularity sequence enum value.

Parameters:

string the string to convert

Returns:

the granularity sequence enum value

Definition at line 140 of file KDChartEnums.h.

```
140                                                                                    {
141       if( string == QString::fromLatin1("GranularitySequence_10_20") )
142           return GranularitySequence_10_20;
143       if( string == QString::fromLatin1("GranularitySequence_10_50") )
144           return GranularitySequence_10_50;
145       if( string == QString::fromLatin1("GranularitySequence_25_50") )
146           return GranularitySequence_25_50;
147       if( string == QString::fromLatin1("GranularitySequence_125") )
148           return GranularitySequence_125_25;
149       if( string == QString::fromLatin1("GranularitySequenceIrregular") )
150           return GranularitySequenceIrregular;
151       // default, should not happen
152       return GranularitySequence_10_20;
153     }
```

7.31.3.6 TextLayoutPolicy KDChartEnums::stringToLayoutPolicy (const QString & *string*) `[static]`

Converts the specified string to a text layout policy enum value.

Parameters:

string the string to convert

Returns:

the text layout policy enum value

7.31.3.7 MeasureCalculationMode KDChartEnums::stringToMeasureCalculationMode (const QString & *string*) `[static]`

Converts the specified string to a measure calculation mode enum value.

Parameters:

string the string to convert

Returns:

the measure calculation mode enum value

Definition at line 267 of file KDChartEnums.h.

```
267                                                                                           {
268       if( string == QString::fromLatin1("MeasureCalculationModeAbsolute") )
269           return MeasureCalculationModeAbsolute;
270       if( string == QString::fromLatin1("MeasureCalculationModeAuto") )
271           return MeasureCalculationModeAuto;
272       if( string == QString::fromLatin1("MeasureCalculationModeAutoArea") )
273           return MeasureCalculationModeAutoArea;
274       if( string == QString::fromLatin1("MeasureCalculationModeAutoOrientation") )
275           return MeasureCalculationModeAutoOrientation;
276       if( string == QString::fromLatin1("MeasureCalculationModeRelative") )
277           return MeasureCalculationModeRelative;
278       // default, should not happen
279       return MeasureCalculationModeAuto;
280     }
```

7.31.3.8 MeasureOrientation KDChartEnums::stringToMeasureOrientation (const QString & *string*) `[static]`

Converts the specified string to a measure orientation enum value.

Parameters:

string the string to convert

Returns:

the measure orientation enum value

Definition at line 336 of file KDChartEnums.h.

```
336                                                                                   {
337       if( string == QString::fromLatin1("MeasureOrientationAuto") )
338           return MeasureOrientationAuto;
339       if( string == QString::fromLatin1("MeasureOrientationHorizontal") )
340           return MeasureOrientationHorizontal;
341       if( string == QString::fromLatin1("MeasureOrientationVertical") )
342           return MeasureOrientationVertical;
343       if( string == QString::fromLatin1("MeasureOrientationMinimum") )
344           return MeasureOrientationMinimum;
345       if( string == QString::fromLatin1("MeasureOrientationMaximum") )
346           return MeasureOrientationMaximum;
347       // default, should not happen
348       return MeasureOrientationAuto;
349     }
```

The documentation for this class was generated from the following file:

- KDChartEnums.h

7.32 KDTextDocument Class Reference

`#include <KDTextDocument.h>`

Inheritance diagram for KDTextDocument:Collaboration diagram for KDTextDocument:

Public Member Functions

- KDTextDocument (const QString &text, QObject ∗parent=0)
- KDTextDocument (QObject ∗parent=0)
- QSize minimumSizeHint ()
- QSize sizeHint ()
- ~KDTextDocument ()

7.32.1 Constructor & Destructor Documentation

7.32.1.1 KDTextDocument::KDTextDocument (QObject ∗ *parent* = 0) `[explicit]`

Definition at line 38 of file KDTextDocument.cpp.

```
39     : QTextDocument( p ),
40       mHintValid( false ),
41       mSizeHint(),
42       mMinimumSizeHint()
43 {
44
45 }
```

7.32.1.2 KDTextDocument::KDTextDocument (const QString & *text*, QObject ∗ *parent* = 0) `[explicit]`

Definition at line 47 of file KDTextDocument.cpp.

```
48     : QTextDocument( text, p ),
49       mHintValid( false ),
50       mSizeHint(),
51       mMinimumSizeHint()
52 {
53
54 }
```

7.32.1.3 KDTextDocument::~KDTextDocument ()

Definition at line 56 of file KDTextDocument.cpp.

```
56 {}
```

7.32.2 Member Function Documentation

7.32.2.1 QSize KDTextDocument::minimumSizeHint ()

Definition at line 66 of file KDTextDocument.cpp.

Referenced by sizeHint().

```
{
    /*
    QTextCursor cursor( this );
    if( ! cursor.atEnd() )
        cursor.movePosition( QTextCursor::NextBlock );
    qDebug() << "KDTextDocument::minimumSizeHint() found:" << cursor.block().text();
    QSizeF s( documentLayout()->blockBoundingRect( cursor.block() ).size() );
    qDebug() << "KDTextDocument::minimumSizeHint() found rect" << documentLayout()->blockBoundingRect(
    return QSize( static_cast<int>(s.width()),
                  static_cast<int>(s.height()) );
    */

    if( mHintValid )
        return mMinimumSizeHint;

    mHintValid = true;
    mSizeHint = sizeForWidth( -1 );
    QSize sz(-1, -1);

    // PENDING(kalle) Cache
    sz.rwidth() = sizeForWidth( 0 ).width();
    sz.rheight() = sizeForWidth( 32000 ).height();
    if( mSizeHint.height() < sz.height())
        sz.rheight() = mSizeHint.height();

    mMinimumSizeHint = sz;
    return sz;
}
```

7.32.2.2 QSize KDTextDocument::sizeHint ()

Definition at line 59 of file KDTextDocument.cpp.

References minimumSizeHint().

```
{
    if( !mHintValid )
        (void)minimumSizeHint();
    return mSizeHint;
}
```

The documentation for this class was generated from the following files:

- KDTextDocument.h
- KDTextDocument.cpp

7.33 KDChart::Legend Class Reference

`#include <KDChartLegend.h>`

Inheritance diagram for KDChart::Legend:Collaboration diagram for KDChart::Legend:

7.33.1 Detailed Description

Legend defines the interface for the legend drawing class.

Legend is the class for drawing legends for all kinds of diagrams ("chart types").

Legend is drawn on chart level, not per diagram, but you can have more than one legend per chart, using KDChart::Chart::addLegend().

Note:

> Legend is different from all other classes ofd KD Chart, since it can be displayed outside of the Chart's area. If you want to, you can embedd the legend into your own widget, or into another part of a bigger grid, into which you might have inserted the Chart.

On the other hand, please note that you MUST call Chart::addLegend to get your legend positioned into the correct place of your chart - if you want to have the legend shown inside of the chart (that's probably true for most cases).

Definition at line 62 of file KDChartLegend.h.

Public Types

- enum LegendStyle {
  MarkersOnly = 0,
  LinesOnly = 1,
  MarkersAndLines = 2 }

Public Member Functions

- void activateTheLayout ()
- void addDiagram (KDChart::AbstractDiagram ∗newDiagram)

  Add the given diagram to the legend.

- Qt::Alignment alignment () const

  Returns the alignment of a non-floating legend.

- void alignToReferencePoint (const RelativePosition &position)
- BackgroundAttributes backgroundAttributes () const
- QBrush brush (uint dataset) const
- const QMap< uint, QBrush > brushes () const
- void buildLegend ()
- virtual Legend ∗ clone () const
- bool compare (const AbstractAreaBase ∗other) const

  Returns true if both areas have the same settings.

- bool compare (const Legend ∗other) const

  Returns true if both axes have the same settings.

- ConstDiagramList constDiagrams () const
- uint datasetCount () const
- KDChart::AbstractDiagram ∗ diagram () const

  The first diagram of the legend or 0 if there was none added to the legend.

- DiagramList diagrams () const

  The list of all diagrams associated with the legend.

- const RelativePosition floatingPosition () const

  Returns the position of a floating legend.

- virtual void forceRebuild ()

  Call this to trigger an unconditional re-building of the widget's internals.

- FrameAttributes frameAttributes () const
- void getFrameLeadings (int &left, int &top, int &right, int &bottom) const
- Legend (KDChart::AbstractDiagram ∗diagram, QWidget ∗parent)
- Legend (QWidget ∗parent=0)
- LegendStyle legendStyle () const
- const QMap< uint, MarkerAttributes > markerAttributes () const
- MarkerAttributes markerAttributes (uint dataset) const
- virtual QSize minimumSizeHint () const
- virtual void needSizeHint ()

  Call this to trigger an conditional re-building of the widget's internals.

- Qt::Orientation orientation () const
- virtual void paint (QPainter ∗painter)

  Overwrite this to paint the inner contents of your widget.

- void paintAll (QPainter &painter)

  Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

- virtual void paintBackground (QPainter &painter, const QRect &rectangle)
- virtual void paintEvent (QPaintEvent ∗event)

  Draws the background and frame, then calls paint().

- virtual void paintFrame (QPainter &painter, const QRect &rectangle)
- virtual void paintIntoRect (QPainter &painter, const QRect &rect)

  Draws the background and frame, then calls paint().

- QPen pen (uint dataset) const
- const QMap< uint, QPen > pens () const
- Position position () const

  Returns the position of a non-floating legend.

- void propertiesChanged ()

Emitted upon change of a property of the Legend or any of its components.

- const QWidget ∗ referenceArea () const

  Returns the reference area, that is used for font size of title text, and for font size of the item texts, IF automatic area detection is set.

- void removeDiagram (KDChart::AbstractDiagram ∗oldDiagram)

  Removes the diagram from the legend's list of diagrams.

- void removeDiagrams ()

  Removes all of the diagram from the legend's list of diagrams.

- void replaceDiagram (KDChart::AbstractDiagram ∗newDiagram, KDChart::AbstractDiagram ∗oldDiagram=0)

  Replaces the old diagram, or appends the new diagram, it there is none yet.

- void resetDiagram (AbstractDiagram ∗)
- void resetTexts ()

  Removes all legend texts that might have been set by setText.

- virtual void resizeEvent (QResizeEvent ∗event)
- virtual void resizeLayout (const QSize &size)
- void setAlignment (Qt::Alignment)

  Specify the alignment of a non-floating legend.

- void setBackgroundAttributes (const BackgroundAttributes &a)
- void setBrush (uint dataset, const QBrush &brush)
- void setBrushesFromDiagram (KDChart::AbstractDiagram ∗diagram)
- void setColor (uint dataset, const QColor &color)

  Note: there is no color() getter method, since setColor just sets a QBrush with the respective color, so the brush() getter method is sufficient.

- void setDefaultColors ()
- void setDiagram (KDChart::AbstractDiagram ∗newDiagram)

  A convenience method doing the same as replaceDiagram(newDiagram, 0);.

- void setFloatingPosition (const RelativePosition &relativePosition)

  Specify the position and alignment of a floating legend.

- void setFrameAttributes (const FrameAttributes &a)
- void setLegendStyle (LegendStyle style)
- void setMarkerAttributes (uint dataset, const MarkerAttributes &)

  Note that any sizes specified via setMarkerAttributes are ignored, unless you disable the automatic size calculation, by saying setUseAutomaticMarkerSize(false).

- void setNeedRebuild ()
- void setOrientation (Qt::Orientation orientation)
- void setPen (uint dataset, const QPen &pen)
- void setPosition (Position position)

  Specify the position of a non-floating legend.

- void setRainbowColors ()
- void setReferenceArea (const QWidget ∗area)

  Specifies the reference area for font size of title text, and for font size of the item texts, IF automatic area detection is set.

- void setShowLines (bool legendShowLines)
- void setSpacing (uint space)
- void setSubduedColors (bool ordered=false)
- void setText (uint dataset, const QString &text)
- void setTextAttributes (const TextAttributes &a)
- void setTitleText (const QString &text)
- void setTitleTextAttributes (const TextAttributes &a)
- void setUseAutomaticMarkerSize (bool useAutomaticMarkerSize)

  This option is on by default, it means that Marker sizes in the Legend will be the same as the font height used for their respective label texts.

- virtual void setVisible (bool visible)
- bool showLines () const
- virtual QSize sizeHint () const
- uint spacing () const
- QString text (uint dataset) const
- TextAttributes textAttributes () const
- const QMap< uint, QString > texts () const
- QString titleText () const
- TextAttributes titleTextAttributes () const
- bool useAutomaticMarkerSize () const
- virtual ∼Legend ()

Static Public Member Functions

- void paintBackgroundAttributes (QPainter &painter, const QRect &rectangle, const KDChart::BackgroundAttributes &attributes)
- void paintFrameAttributes (QPainter &painter, const QRect &rectangle, const KDChart::FrameAttributes &attributes)

Public Attributes

- Q_SIGNALS __pad0__: void destroyedLegend(Legend∗)
- private Q_SLOTS: void emitPositionChanged()

Protected Member Functions

- virtual QRect areaGeometry () const
- QRect innerRect () const
- virtual void positionHasChanged ()

7.33.2 Member Enumeration Documentation

7.33.2.1 enum KDChart::Legend::LegendStyle

Enumeration values:

MarkersOnly

LinesOnly

MarkersAndLines

Definition at line 75 of file KDChartLegend.h.

Referenced by buildLegend().

```
75                        { MarkersOnly     = 0,
76                          LinesOnly       = 1,
77                          MarkersAndLines = 2 };
```

7.33.3 Constructor & Destructor Documentation

7.33.3.1 Legend::Legend (QWidget ∗ *parent* = 0) `[explicit]`

Definition at line 85 of file KDChartLegend.cpp.

References d.

Referenced by clone().

```
85                                 :
86       AbstractAreaWidget( new Private(), parent )
87 {
88       d->referenceArea = parent;
89       init();
90 }
```

7.33.3.2 Legend::Legend (KDChart::AbstractDiagram ∗ *diagram*, QWidget ∗ *parent*) `[explicit]`

Definition at line 92 of file KDChartLegend.cpp.

References d, and setDiagram().

```
92                                                                  :
93       AbstractAreaWidget( new Private(), parent )
94 {
95       d->referenceArea = parent;
96       init();
97       setDiagram( diagram );
98 }
```

7.33.3.3 Legend::∼Legend () `[virtual]`

Definition at line 100 of file KDChartLegend.cpp.

```
{
    emit destroyedLegend( this );
}
```

7.33.4 Member Function Documentation

7.33.4.1 void Legend::activateTheLayout ()

Definition at line 182 of file KDChartLegend.cpp.

References d.

Referenced by buildLegend(), and resizeLayout().

```
{
    if( d->layout && d->layout->parent() )
        d->layout->activate();
}
```

7.33.4.2 void Legend::addDiagram (KDChart::AbstractDiagram ∗ *newDiagram*)

Add the given diagram to the legend.

Parameters:

newDiagram The diagram to add.

See also:

diagram, diagrams, removeDiagram, removeDiagrams, replaceDiagram, setDiagram

Definition at line 334 of file KDChartLegend.cpp.

References d, resetDiagram(), and setNeedRebuild().

Referenced by replaceDiagram().

```
{
    if ( newDiagram )
    {
        DiagramObserver* observer = new DiagramObserver( newDiagram, this );

        DiagramObserver* oldObs = d->findObserverForDiagram( newDiagram );
        if( oldObs ){
            delete oldObs;
            d->observers[ d->observers.indexOf( oldObs ) ] = observer;
        }else{
            d->observers.append( observer );
        }
        connect( observer, SIGNAL( diagramDestroyed(AbstractDiagram*) ),
                        SLOT( resetDiagram(AbstractDiagram*) ));
        connect( observer, SIGNAL( diagramDataChanged(AbstractDiagram*) ),
                 SLOT( setNeedRebuild() ));
        connect( observer, SIGNAL( diagramDataHidden(AbstractDiagram*) ),
                 SLOT( setNeedRebuild() ));
        connect( observer, SIGNAL( diagramAttributesChanged(AbstractDiagram*) ),
                        SLOT( setNeedRebuild() ));
        setNeedRebuild();
    }
}
```

7.33.4.3 Qt::Alignment Legend::alignment () const

Returns the alignment of a non-floating legend.

See also:
setAlignment

Definition at line 443 of file KDChartLegend.cpp.

References d.

Referenced by clone().

```
{
    return d->alignment;
}
```

7.33.4.4 void AbstractAreaBase::alignToReferencePoint (const RelativePosition & *position*) `[inherited]`

Definition at line 90 of file KDChartAbstractAreaBase.cpp.

```
{
    Q_UNUSED( position );
    // PENDING(kalle) FIXME
    qWarning( "Sorry, not implemented: void AbstractAreaBase::alignToReferencePoint( const RelativePosi
}
```

7.33.4.5 QRect AbstractAreaWidget::areaGeometry () const `[protected, virtual, inherited]`

Implements KDChart::AbstractAreaBase.

Definition at line 186 of file KDChartAbstractAreaWidget.cpp.

```
{
    return geometry();
}
```

7.33.4.6 BackgroundAttributes AbstractAreaBase::backgroundAttributes () const `[inherited]`

Definition at line 112 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by updateCommonBrush().

```
{
    return d->backgroundAttributes;
}
```

7.33.4.7 QBrush Legend::brush (uint *dataset*) const

Definition at line 549 of file KDChartLegend.cpp.

References d.

Referenced by buildLegend(), and setRainbowColors().

```
{
    if( d->brushes.find( dataset ) != d->brushes.end() )
        return d->brushes[ dataset ];
    else
        return d->modelBrushes[ dataset ];
}
```

7.33.4.8 const QMap< uint, QBrush > Legend::brushes () const

Definition at line 557 of file KDChartLegend.cpp.

References d.

```
{
    return d->brushes;
}
```

7.33.4.9 void Legend::buildLegend ()

Definition at line 770 of file KDChartLegend.cpp.

References activateTheLayout(), brush(), KDChart::TextAttributes::calculatedFontSize(), d, KDChart::AbstractDiagram::datasetBrushes(), KDChart::AbstractDiagram::datasetLabels(), KDChart::AbstractDiagram::datasetMarkers(), KDChart::AbstractDiagram::datasetPens(), diagram(), KDChart::AbstractDiagram::isHidden(), KDChart::TextAttributes::isVisible(), legendStyle(), LegendStyle, LinesOnly, markerAttributes(), MarkersAndLines, MarkersOnly, orientation(), pen(), propertiesChanged(), referenceArea(), KDChart::AbstractLayoutItem::setParentWidget(), showLines(), spacing(), text(), textAttributes(), titleText(), titleTextAttributes(), and useAutomaticMarkerSize().

Referenced by forceRebuild(), needSizeHint(), and setNeedRebuild().

```
{
    /*
    if( !d->needRebuild ) {
#ifdef DEBUG_LEGEND_PAINT
        qDebug() << "leaving Legend::buildLegend() with NO action (was already build)";
#endif
        // Note: We do *not* need to send positionChanged here,
        //       because we send it in the resizeEvent, so layouting
        //       is done at the right time.
        return;
    }
#ifdef DEBUG_LEGEND_PAINT
    qDebug() << "entering Legend::buildLegend() **********************************";
#endif
    d->needRebuild = false;
    */

    Q_FOREACH( QLayoutItem* layoutItem, d->layoutItems ) {
        d->layout->removeItem( layoutItem );
    }
    qDeleteAll( d->layoutItems );
    d->layoutItems.clear();

    if( orientation() == Qt::Vertical ) {
        d->layout->setColumnStretch( 4, 1 );
    } else {
        d->layout->setColumnStretch( 4, 0 );
    }

    d->modelLabels.clear();
    d->modelBrushes.clear();
    d->modelPens.clear();
    d->modelMarkers.clear();
    // retrieve the diagrams' settings for all non-hidden datasets
    for (int i = 0; i < d->observers.size(); ++i){
        const AbstractDiagram* diagram = d->observers.at(i)->diagram();
        if( diagram ){
            //qDebug() << "accessing" << diagram;
            const QStringList             diagramLabels(  diagram->datasetLabels()  );
            const QList<QBrush>           diagramBrushes( diagram->datasetBrushes() );
            const QList<QPen>             diagramPens(    diagram->datasetPens()    );
            const QList<MarkerAttributes> diagramMarkers( diagram->datasetMarkers() );
            for ( int dataset = 0; dataset < diagramLabels.count(); dataset++ ) {
                // only show the label if the diagrams is NOT having the dataset set to hidden
                if( ! diagram->isHidden( dataset ) ){
                    d->modelLabels  += diagramLabels[   dataset ];
                    d->modelBrushes += diagramBrushes[  dataset ];
                    d->modelPens    += diagramPens[     dataset ];
                    d->modelMarkers += diagramMarkers[  dataset ];
                }
            }
        }
    }

    Q_ASSERT( d->modelLabels.count() == d->modelBrushes.count() );

    // legend caption
    if( !titleText().isEmpty() && titleTextAttributes().isVisible() ) {
        // PENDING(kalle) Other properties!
        KDChart::TextLayoutItem* titleItem =
            new KDChart::TextLayoutItem( titleText(),
                titleTextAttributes(),
                referenceArea(),
                (orientation() == Qt::Vertical)
                ? KDChartEnums::MeasureOrientationMinimum
                : KDChartEnums::MeasureOrientationHorizontal,
                Qt::AlignCenter );
        titleItem->setParentWidget( this );

        d->layoutItems << titleItem;
        if( orientation() == Qt::Vertical )
            d->layout->addItem( titleItem, 0, 0, 1, 5, Qt::AlignCenter );
        else
            d->layout->addItem( titleItem, 0, 0, 1, d->modelLabels.count() ? (d->modelLabels.count()*4

        // The line between the title and the legend items, if any.
        if( showLines() && d->modelLabels.count() ) {
            KDChart::HorizontalLineLayoutItem* lineItem = new KDChart::HorizontalLineLayoutItem();
            d->layoutItems << lineItem;
            if( orientation() == Qt::Vertical )
                d->layout->addItem( lineItem, 1, 0, 1, 5, Qt::AlignCenter );
            else
                d->layout->addItem( lineItem, 1, 0, 1, d->modelLabels.count()*4, Qt::AlignCenter );
        }
    }

    const KDChartEnums::MeasureOrientation orient =
            (orientation() == Qt::Vertical)
            ? KDChartEnums::MeasureOrientationMinimum
            : KDChartEnums::MeasureOrientationHorizontal;
    const TextAttributes labelAttrs( textAttributes() );
    const qreal fontHeight = labelAttrs.calculatedFontSize( referenceArea(), orient );
    const LegendStyle style = legendStyle();
    //qDebug() << "fontHeight:" << fontHeight;

    const bool bShowMarkers = (style != LinesOnly);

    QSizeF maxMarkersSize(1.0, 1.0);
    QVector <MarkerAttributes> markerAttrs( d->modelLabels.count() );
    if( bShowMarkers ){
        for ( int dataset = 0; dataset < d->modelLabels.count(); ++dataset ) {
            markerAttrs[dataset] = markerAttributes( dataset );
            QSizeF siz;
            if( useAutomaticMarkerSize() ||
                ! markerAttrs[dataset].markerSize().isValid() )
            {
                siz = QSizeF(fontHeight, fontHeight);
                markerAttrs[dataset].setMarkerSize( siz );
            }else{
                siz = markerAttrs[dataset].markerSize();
            }
            maxMarkersSize =
                    QSizeF(qMax(maxMarkersSize.width(),  siz.width()),
                           qMax(maxMarkersSize.height(), siz.height()));
        }
    }

    // If we show a marker on a line, we paint it after 4 pixels
    // of the line have been painted. This allows to see the line style
    // at the right side of the marker without the line needing to
    // be too long.
    // (having the marker in the middle of the line would require longer lines)
    const int markerOffsOnLine = 8;

    int maxLineLength = 18;
    {
        bool hasComplexPenStyle = false;
        for ( int dataset = 0; dataset < d->modelLabels.count(); ++dataset ){
            const QPen pn = pen(dataset);
            const Qt::PenStyle ps = pn.style();
            if( ps != Qt::NoPen ){
                maxLineLength = qMax( pn.width() * 18, maxLineLength );
                if( ps != Qt::SolidLine )
                    hasComplexPenStyle = true;
            }
        }
        if( hasComplexPenStyle && bShowMarkers )
            maxLineLength =
                    maxLineLength + markerOffsOnLine +
                    static_cast<int>(maxMarkersSize.width());
    }

    for ( int dataset = 0; dataset < d->modelLabels.count(); ++dataset ) {
        KDChart::AbstractLayoutItem* markerLineItem = 0;
        switch( style ){
            case( MarkersOnly ):
                markerLineItem = new KDChart::MarkerLayoutItem(
                        diagram(),
                        markerAttrs[dataset],
                        brush( dataset ),
                        pen( dataset ),
                        Qt::AlignLeft );
                break;
            case( LinesOnly ):
                markerLineItem = new KDChart::LineLayoutItem(
                        diagram(),
                        maxLineLength,
                        pen( dataset ),
                        Qt::AlignCenter );
                break;
            case( MarkersAndLines ):
                markerLineItem = new KDChart::LineWithMarkerLayoutItem(
                        diagram(),
                        maxLineLength,
                        pen( dataset ),
                        markerOffsOnLine,
                        markerAttrs[dataset],
                        brush( dataset ),
                        pen( dataset ),
                        Qt::AlignCenter );
                break;
            default:
                Q_ASSERT( false ); // all styles need to be handled
        }
        if( markerLineItem ){
            d->layoutItems << markerLineItem;
            if( orientation() == Qt::Vertical )
                d->layout->addItem( markerLineItem,
                                    dataset*2+2, // first row is title, second is line
                                    1,
                                    1, 1, Qt::AlignCenter );
            else
                d->layout->addItem( markerLineItem,
                                    2, // all in row two
                                    dataset*4 );
        }

        // PENDING(kalle) Other properties!
        KDChart::TextLayoutItem* labelItem =
            new KDChart::TextLayoutItem( text( dataset ),
                labelAttrs,
                referenceArea(), orient,
                Qt::AlignLeft );
        labelItem->setParentWidget( this );

        d->layoutItems << labelItem;
        if( orientation() == Qt::Vertical )
            d->layout->addItem( labelItem,
                                dataset*2+2, // first row is title, second is line
                                3 );
        else
            d->layout->addItem( labelItem,
                                2, // all in row two
                                dataset*4+1 );

        // horizontal lines (only in vertical mode, and not after the last item)
        if( orientation() == Qt::Vertical && showLines() && dataset != d->modelLabels.count()-1 ) {
            KDChart::HorizontalLineLayoutItem* lineItem = new KDChart::HorizontalLineLayoutItem();
            d->layoutItems << lineItem;
            d->layout->addItem( lineItem,
                                dataset*2+1+2,
                                0,
                                1, 5, Qt::AlignCenter );
        }

        // vertical lines (only in horizontal mode, and not after the last item)
        if( orientation() == Qt::Horizontal && showLines() && dataset != d->modelLabels.count()-1 ) {
            KDChart::VerticalLineLayoutItem* lineItem = new KDChart::VerticalLineLayoutItem();
            d->layoutItems << lineItem;
            d->layout->addItem( lineItem,
                                 2, // all in row two
                                style == MarkersAndLines ? dataset*4+3 : dataset*4+2 ,
                                1, 1, Qt::AlignCenter );
        }

        if( orientation() != Qt::Vertical ) { // Horizontal needs a spacer
            d->layout->addItem( new QSpacerItem( spacing(), 1 ),
                                2, // all in row two
                                dataset*4+3 );
         }
     }

     // vertical line (only in vertical mode)
     if( orientation() == Qt::Vertical && showLines() && d->modelLabels.count() ) {
         KDChart::VerticalLineLayoutItem* lineItem = new KDChart::VerticalLineLayoutItem();
         d->layoutItems << lineItem;
         d->layout->addItem( lineItem, 2, 2, d->modelLabels.count()*2, 1 );
     }

     // This line is absolutely necessary, otherwise: #2516.
     activateTheLayout();

     emit propertiesChanged();
     //emit positionChanged( this );
     //emitPositionChanged();
 #ifdef DEBUG_LEGEND_PAINT
     qDebug() << "leaving Legend::buildLegend()";
 #endif
 }
```

7.33.4.10 Legend * Legend::clone () const `[virtual]`

Definition at line 201 of file KDChartLegend.cpp.

References alignment(), d, KDChart::AbstractAreaBase::frameAttributes(), Legend(), legendStyle(), position(), setAlignment(), KDChart::AbstractAreaBase::setFrameAttributes(), setLegendStyle(), setPosition(), setTextAttributes(), setTitleTextAttributes(), setUseAutomaticMarkerSize(), textAttributes(), titleTextAttributes(), and useAutomaticMarkerSize().

```
{
    Legend* legend = new Legend( new Private( *d ), 0 );
    legend->setTextAttributes( textAttributes() );
    legend->setTitleTextAttributes( titleTextAttributes() );
    legend->setFrameAttributes( frameAttributes() );
    legend->setUseAutomaticMarkerSize( useAutomaticMarkerSize() );
    legend->setPosition( position() );
    legend->setAlignment( alignment() );
    legend->setLegendStyle( legendStyle() );
    return legend;
}
```

7.33.4.11 bool AbstractAreaBase::compare (const AbstractAreaBase * *other*) const `[inherited]`

Returns true if both areas have the same settings.

Definition at line 75 of file KDChartAbstractAreaBase.cpp.

```
{
    if( other == this ) return true;
    if( ! other ){
        //qDebug() << "CartesianAxis::compare() cannot compare to Null pointer";
        return false;
    }
    /*
    qDebug() << "AbstractAreaBase:" << (frameAttributes() == other->frameAttributes())
        << (backgroundAttributes() == other->backgroundAttributes()) << "\n";
    */
    return  (frameAttributes()      == other->frameAttributes()) &&
            (backgroundAttributes() == other->backgroundAttributes());
}
```

7.33.4.12 bool Legend::compare (const Legend ∗ *other*) const

Returns true if both axes have the same settings.

Definition at line 215 of file KDChartLegend.cpp.

```
{
    if( other == this ) return true;
    if( ! other ){
        //qDebug() << "Legend::compare() cannot compare to Null pointer";
        return false;
    }
    /*
    qDebug() << ( static_cast<const AbstractAreaBase*>(this)->compare( other ) );
    qDebug() << (isVisible()              == other->isVisible());
    qDebug() << (position()               == other->position());
    qDebug() << (alignment()              == other->alignment());
    qDebug() << (floatingPosition()       == other->floatingPosition());
    qDebug() << (orientation()            == other->orientation());
    qDebug() << (showLines()              == other->showLines());
    qDebug() << (texts()                  == other->texts());
    qDebug() << (brushes()                == other->brushes());
    qDebug() << (pens()                   == other->pens());
    qDebug() << (markerAttributes()       == other->markerAttributes());
    qDebug() << (useAutomaticMarkerSize() == other->useAutomaticMarkerSize());
    qDebug() << (textAttributes()         == other->textAttributes());
    qDebug() << (titleText()              == other->titleText());
    qDebug() << (titleTextAttributes()    == other->titleTextAttributes());
    qDebug() << (spacing()                == other->spacing());
    qDebug() << (legendStyle()            == other->legendStyle());
    */
    return  ( static_cast<const AbstractAreaBase*>(this)->compare( other ) ) &&
            (isVisible()              == other->isVisible()) &&
            (position()               == other->position()) &&
            (alignment()              == other->alignment())&&
            (floatingPosition()       == other->floatingPosition()) &&
            (orientation()            == other->orientation())&&
            (showLines()              == other->showLines())&&
            (texts()                  == other->texts())&&
            (brushes()                == other->brushes())&&
            (pens()                   == other->pens())&&
            (markerAttributes()       == other->markerAttributes())&&
            (useAutomaticMarkerSize() == other->useAutomaticMarkerSize()) &&
            (textAttributes()         == other->textAttributes()) &&
            (titleText()              == other->titleText())&&
            (titleTextAttributes()    == other->titleTextAttributes()) &&
            (spacing()                == other->spacing()) &&
            (legendStyle()            == other->legendStyle());
}
```

7.33.4.13 ConstDiagramList Legend::constDiagrams () const

Returns:

The list of diagrams associated with this coordinate plane.

Definition at line 326 of file KDChartLegend.cpp.

References KDChart::ConstDiagramList, and d.

```
{
    ConstDiagramList list;
    for (int i = 0; i < d->observers.size(); ++i)
        list << d->observers.at(i)->diagram();
    return list;
}
```

7.33.4.14 uint Legend::datasetCount () const

Definition at line 283 of file KDChartLegend.cpp.

References d, KDChart::AbstractDiagram::datasetBrushes(), KDChart::AbstractDiagram::datasetLabels(), and KDChart::DiagramObserver::diagram().

```
{
    int modelLabelsCount = 0;
    int modelBrushesCount = 0;
    for (int i = 0; i < d->observers.size(); ++i) {
        DiagramObserver * obs = d->observers.at(i);
        modelLabelsCount  += obs->diagram()->datasetLabels().count();
        modelBrushesCount += obs->diagram()->datasetBrushes().count();
    }
    Q_ASSERT( modelLabelsCount == modelBrushesCount );
    return modelLabelsCount;
}
```

7.33.4.15 AbstractDiagram ∗ Legend::diagram () const

The first diagram of the legend or 0 if there was none added to the legend.

Returns:

The first diagram of the legend or 0.

See also:

diagrams, addDiagram, removeDiagram, removeDiagrams, replaceDiagram, setDiagram

Definition at line 311 of file KDChartLegend.cpp.

References d.

Referenced by buildLegend(), and paint().

```
{
    if( d->observers.isEmpty() )
        return 0;
    return d->observers.first()->diagram();
}
```

7.33.4.16 DiagramList Legend::diagrams () const

The list of all diagrams associated with the legend.

Returns:
The list of all diagrams associated with the legend.

See also:
diagram, addDiagram, removeDiagram, removeDiagrams, replaceDiagram, setDiagram

Definition at line 318 of file KDChartLegend.cpp.

References d, and KDChart::DiagramList.

```
319 {
320     DiagramList list;
321     for (int i = 0; i < d->observers.size(); ++i)
322         list << d->observers.at(i)->diagram();
323     return list;
324 }
```

7.33.4.17 const RelativePosition Legend::floatingPosition () const

Returns the position of a floating legend.

See also:
setFloatingPosition

Definition at line 457 of file KDChartLegend.cpp.

References d.

Referenced by KDChart::Chart::reLayoutFloatingLegends().

```
458 {
459     return d->relativePosition;
460 }
```

7.33.4.18 void Legend::forceRebuild () `[virtual]`

Call this to trigger an unconditional re-building of the widget’s internals.

Reimplemented from KDChart::AbstractAreaWidget.

Definition at line 657 of file KDChartLegend.cpp.

References buildLegend().

Referenced by resizeEvent().

```
658 {
659 #ifdef DEBUG_LEGEND_PAINT
660     qDebug() << "entering Legend::forceRebuild()";
661 #endif
662     //setSpacing(d->layout->spacing());
663     buildLegend();
664 #ifdef DEBUG_LEGEND_PAINT
665     qDebug() << "leaving Legend::forceRebuild()";
666 #endif
667 }
```

7.33.4.19 FrameAttributes AbstractAreaBase::frameAttributes () const `[inherited]`

Definition at line 102 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by clone(), and updateCommonBrush().

```
{
    return d->frameAttributes;
}
```

7.33.4.20 void AbstractAreaBase::getFrameLeadings (int & *left*, int & *top*, int & *right*, int & *bottom*) const `[inherited]`

Definition at line 204 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::AbstractAreaBase::innerRect(), and KDChart::AbstractAreaWidget::paintAll().

```
{
    if( d && d->frameAttributes.isVisible() ){
        const int padding = qMax( d->frameAttributes.padding(), 0 );
        left   = padding;
        top    = padding;
        right  = padding;
        bottom = padding;
    }else{
        left   = 0;
        top    = 0;
        right  = 0;
        bottom = 0;
    }
}
```

7.33.4.21 QRect AbstractAreaBase::innerRect () const `[protected, inherited]`

Definition at line 220 of file KDChartAbstractAreaBase.cpp.

References KDChart::AbstractAreaBase::areaGeometry(), and KDChart::AbstractAreaBase::getFrameLeadings().

Referenced by KDChart::TextArea::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    int left;
    int top;
    int right;
    int bottom;
    getFrameLeadings( left, top, right, bottom );
    return
        QRect( QPoint(0,0), areaGeometry().size() )
            .adjusted( left, top, -right, -bottom );
}
```

7.33.4.22 Legend::LegendStyle Legend::legendStyle () const

Definition at line 196 of file KDChartLegend.cpp.

References d.

Referenced by buildLegend(), and clone().

```
{
    return d->legendStyle;
}
```

7.33.4.23 const QMap< uint, MarkerAttributes > Legend::markerAttributes () const

Definition at line 615 of file KDChartLegend.cpp.

References d.

Referenced by buildLegend().

```
{
    return d->markerAttributes;
}
```

7.33.4.24 MarkerAttributes Legend::markerAttributes (uint *dataset*) const

Definition at line 606 of file KDChartLegend.cpp.

References d.

```
{
    if( d->markerAttributes.find( dataset ) != d->markerAttributes.end() )
        return d->markerAttributes[ dataset ];
    else if ( static_cast<uint>( d->modelMarkers.count() ) > dataset )
        return d->modelMarkers[ dataset ];
    return MarkerAttributes();
}
```

7.33.4.25 QSize Legend::minimumSizeHint () const `[virtual]`

Definition at line 143 of file KDChartLegend.cpp.

References sizeHint().

```
{
    return sizeHint();
}
```

7.33.4.26 void Legend::needSizeHint () `[virtual]`

Call this to trigger an conditional re-building of the widget's internals.

e.g. AbstractAreaWidget call this, before calling layout()->setGeometry()

Reimplemented from KDChart::AbstractAreaWidget.

Definition at line 161 of file KDChartLegend.cpp.

References buildLegend().

```
{
    // Re-build the Legend's content, if it has not been build yet,
    // or if the Legend's geometry has changed, resp.
    buildLegend();
}
```

7.33.4.27 Qt::Orientation Legend::orientation () const

Definition at line 470 of file KDChartLegend.cpp.

References d.

Referenced by buildLegend().

```
{
    return d->orientation;
}
```

7.33.4.28 void Legend::paint (QPainter ∗ *painter*) `[virtual]`

Overwrite this to paint the inner contents of your widget.

Note:

When overriding this method, please let your widget draw itself at the top/left corner of the painter. You should call rect() (or width(), height(), resp.) to find the drawable area's size: While the paint() method is being executed the frame of the widget is outside of its rect(), so you can use all of rect() for your custom drawing!

See also:

paint, paintIntoRect

Implements KDChart::AbstractAreaWidget.

Definition at line 261 of file KDChartLegend.cpp.

References d, diagram(), and KDChart::AbstractLayoutItem::paint().

```
{
#ifdef DEBUG_LEGEND_PAINT
    qDebug() << "entering Legend::paint( QPainter* painter )";
#endif
    // rule: We do not show a legend, if there is no diagram.
    if( ! diagram() ) return;

    // re-calculate/adjust the Legend's internal layout and contents, if needed:
    //buildLegend();

    // PENDING(kalle) Support palette

    Q_FOREACH( KDChart::AbstractLayoutItem* layoutItem, d->layoutItems ) {
        layoutItem->paint( painter );
    }
#ifdef DEBUG_LEGEND_PAINT
    qDebug() << "leaving Legend::paint( QPainter* painter )";
#endif
}
```

7.33.4.29 void AbstractAreaWidget::paintAll (QPainter & *painter*) `[inherited]`

Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

Definition at line 145 of file KDChartAbstractAreaWidget.cpp.

References KDChart::AbstractAreaBase::getFrameLeadings(), KDChart::AbstractAreaWidget::paint(), KDChart::AbstractAreaBase::paintBackground(), and KDChart::AbstractAreaBase::paintFrame().

Referenced by KDChart::AbstractAreaWidget::paintEvent(), and KDChart::AbstractAreaWidget::paintIntoRect().

```
{
    //qDebug() << "AbstractAreaWidget::paintAll() called";

    // Paint the background and frame
    paintBackground( painter, QRect(QPoint(0, 0), size() ) );
    paintFrame(      painter, QRect(QPoint(0, 0), size() ) );

/*
    we do not call setContentsMargins() now,
    but we call resizeLayout() whenever the size or the frame has changed

    // adjust the widget’s content margins,
    // to be sure all content gets calculated
    // to fit into the inner rectangle
    const QRect oldGeometry( areaGeometry()  );
    const QRect inner( innerRect() );
    //qDebug() << "areaGeometry():" << oldGeometry
    //         << "  contentsRect():" << contentsRect() << "  inner:" << inner;
    if( contentsRect() != inner ){
        //qDebug() << "old contentsRect():" << contentsRect() << "  new innerRect:" << inner;
        setContentsMargins(
            inner.left(),
            inner.top(),
            oldGeometry.width() -inner.width()-1,
            oldGeometry.height()-inner.height()-1 );
        //forceRebuild();
    }
*/
    int left;
    int top;
    int right;
    int bottom;
    getFrameLeadings( left, top, right, bottom );
    const QPoint translation( left, top );
    painter.translate( translation );
    paint( &painter );
    painter.translate( -translation.x(), -translation.y() );
     //qDebug() << "AbstractAreaWidget::paintAll() done.";
}
```

7.33.4.30 void AbstractAreaBase::paintBackground (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 188 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintBackgroundAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintBackground()",
                "Private class was not initialized!" );
    paintBackgroundAttributes( painter, rect, d->backgroundAttributes );
}
```

7.33.4.31 void AbstractAreaBase::paintBackgroundAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::BackgroundAttributes & *attributes*) `[static, inherited]`

Definition at line 119 of file KDChartAbstractAreaBase.cpp.

References KDChart::BackgroundAttributes::brush(), KDChart::BackgroundAttributes::isVisible(), KDChart::BackgroundAttributes::pixmap(), and KDChart::BackgroundAttributes::pixmapMode().

Referenced by KDChart::AbstractAreaBase::paintBackground().

```
{
    if( !attributes.isVisible() ) return;

    /* first draw the brush (may contain a pixmap)*/
    if( Qt::NoBrush != attributes.brush().style() ) {
        KDChart::PainterSaver painterSaver( &painter );
        painter.setPen( Qt::NoPen );
        const QPointF newTopLeft( painter.deviceMatrix().map( rect.topLeft() ) );
        painter.setBrushOrigin( newTopLeft );
        painter.setBrush( attributes.brush() );
        painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
    }
    /* next draw the backPixmap over the brush */
    if( !attributes.pixmap().isNull() &&
        attributes.pixmapMode() != BackgroundAttributes::BackgroundPixmapModeNone ) {
        QPointF ol = rect.topLeft();
        if( BackgroundAttributes::BackgroundPixmapModeCentered == attributes.pixmapMode() )
        {
            ol.setX( rect.center().x() - attributes.pixmap().width() / 2 );
            ol.setY( rect.center().y() - attributes.pixmap().height()/ 2 );
            painter.drawPixmap( ol, attributes.pixmap() );
        } else {
            QMatrix m;
            double zW = (double)rect.width()  / (double)attributes.pixmap().width();
            double zH = (double)rect.height() / (double)attributes.pixmap().height();
            switch( attributes.pixmapMode() ) {
            case BackgroundAttributes::BackgroundPixmapModeScaled:
            {
                double z;
                z = qMin( zW, zH );
                m.scale( z, z );
            }
            break;
            case BackgroundAttributes::BackgroundPixmapModeStretched:
                m.scale( zW, zH );
156                 break;
157             default:
158                 ; // Cannot happen, previously checked
159             }
160             QPixmap pm = attributes.pixmap().transformed( m );
161             ol.setX( rect.center().x() - pm.width() / 2 );
162             ol.setY( rect.center().y() - pm.height()/ 2 );
163             painter.drawPixmap( ol, pm );
164         }
165     }
166 }
```

7.33.4.32 void AbstractAreaWidget::paintEvent (QPaintEvent ∗ *event*) `[virtual, inherited]`

Draws the background and frame, then calls paint().

In most cases there is no need to overwrite this method in a derived class, but you would overwrite paint() instead.

See also:

paint

Definition at line 99 of file KDChartAbstractAreaWidget.cpp.

References d, and KDChart::AbstractAreaWidget::paintAll().

```
100 {
101     Q_UNUSED( event );
102     QPainter painter( this );
103     if( size() != d->currentLayoutSize ){
104         d->resizeLayout( this, size() );
105     }
106     paintAll( painter );
107 }
```

7.33.4.33 void AbstractAreaBase::paintFrame (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 196 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintFrameAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
197 {
198     Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintFrame()",
199                 "Private class was not initialized!" );
200     paintFrameAttributes( painter, rect, d->frameAttributes );
201 }
```

7.33.4.34 void AbstractAreaBase::paintFrameAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::FrameAttributes & *attributes*) `[static, inherited]`

Definition at line 169 of file KDChartAbstractAreaBase.cpp.

References KDChart::FrameAttributes::isVisible(), and KDChart::FrameAttributes::pen().

Referenced by KDChart::AbstractAreaBase::paintFrame().

```
{

    if( !attributes.isVisible() ) return;

    // Note: We set the brush to NoBrush explicitely here.
    //       Otherwise we might get a filled rectangle, so any
    //       previously drawn background would be overwritten by that area.

    const QPen   oldPen(   painter.pen() );
    const QBrush oldBrush( painter.brush() );
    painter.setPen(   attributes.pen() );
    painter.setBrush( Qt::NoBrush );
    painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
    painter.setBrush( oldBrush );
    painter.setPen(   oldPen );
}
```

7.33.4.35 void AbstractAreaWidget::paintIntoRect (QPainter & *painter*, const QRect & *rect*) `[virtual, inherited]`

Draws the background and frame, then calls paint().

In most cases there is no need to overwrite this method in a derived class, but you would overwrite paint() instead.

Definition at line 109 of file KDChartAbstractAreaWidget.cpp.

References d, and KDChart::AbstractAreaWidget::paintAll().

Referenced by KDChart::Chart::paint().

```
{
    //qDebug() << "AbstractAreaWidget::paintIntoRect() called rect=" << rect;

    if( rect.isEmpty() ) return;

    d->resizeLayout( this, rect.size() );

    const QPoint translation( rect.topLeft() );
    painter.translate( translation );
    paintAll( painter );
    painter.translate( -translation.x(), -translation.y() );

/*
    // make sure, the contents of the widget have been set up,
    // so we get a usefull geometry:
    needSizeHint();

    const QRect oldGeometry( layout()->geometry() );
    const QRect newGeo( QPoint(0,0), rect.size() );
    const bool mustChangeGeo = layout() && oldGeometry != newGeo;
    if( mustChangeGeo )
        layout()->setGeometry( newGeo );
    painter.translate( rect.left(), rect.top() );
    paintAll( painter );
    painter.translate( -rect.left(), -rect.top() );
    if( mustChangeGeo )
        layout()->setGeometry( oldGeometry );
*/
}
```

7.33.4.36 QPen Legend::pen (uint *dataset*) const

Definition at line 585 of file KDChartLegend.cpp.

References d.

Referenced by buildLegend().

```
586 {
587     if( d->pens.find( dataset ) != d->pens.end() )
588         return d->pens[dataset];
589     else
590         return d->modelPens[ dataset ];
591 }
```

7.33.4.37 const QMap< uint, QPen > Legend::pens () const

Definition at line 593 of file KDChartLegend.cpp.

References d.

```
594 {
595     return d->pens;
596 }
```

7.33.4.38 Position Legend::position () const

Returns the position of a non-floating legend.

See also:
setPosition

Definition at line 432 of file KDChartLegend.cpp.

References d.

Referenced by clone(), and KDChart::Chart::reLayoutFloatingLegends().

```
433 {
434     return d->position;
435 }
```

7.33.4.39 void AbstractAreaWidget::positionHasChanged () `[protected, virtual, inherited]`

Reimplemented from KDChart::AbstractAreaBase.

Definition at line 191 of file KDChartAbstractAreaWidget.cpp.

```
192 {
193     emit positionChanged( this );
194 }
```

7.33.4.40 void KDChart::Legend::propertiesChanged ()

Emitted upon change of a property of the Legend or any of its components.

Referenced by buildLegend().

7.33.4.41 const QWidget ∗ Legend::referenceArea () const

Returns the reference area, that is used for font size of title text, and for font size of the item texts, IF automatic area detection is set.

See also:

setReferenceArea

Definition at line 304 of file KDChartLegend.cpp.

References d.

Referenced by buildLegend().

```
{
    //qDebug() << d->referenceArea;
    return (d->referenceArea ? d->referenceArea : static_cast<const QWidget*>(parent()));
}
```

7.33.4.42 void Legend::removeDiagram (KDChart::AbstractDiagram ∗ *oldDiagram*)

Removes the diagram from the legend's list of diagrams.

See also:

diagram, diagrams, addDiagram, removeDiagrams, replaceDiagram, setDiagram

Definition at line 359 of file KDChartLegend.cpp.

References d, and setNeedRebuild().

Referenced by removeDiagrams(), replaceDiagram(), and resetDiagram().

```
{
    if( oldDiagram ){
        DiagramObserver* oldObs = d->findObserverForDiagram( oldDiagram );
        if( oldObs ){
            //qDebug() << "before delete oldObs;";
            delete oldObs;
            //qDebug() << "after delete oldObs;";
            d->observers.removeAt( d->observers.indexOf( oldObs ) );
            //qDebug() << "after d->observers.removeAt()";
        }
        setNeedRebuild();
    }
}
```

7.33.4.43 void Legend::removeDiagrams ()

Removes all of the diagram from the legend's list of diagrams.

See also:

diagram, diagrams, addDiagram, removeDiagram, replaceDiagram, setDiagram

Definition at line 374 of file KDChartLegend.cpp.

References d, and removeDiagram().

```
{
    for (int i = 0; i < d->observers.size(); ++i)
        removeDiagram( d->observers.at(i)->diagram() );
}
```

7.33.4.44 void Legend::replaceDiagram (KDChart::AbstractDiagram * *newDiagram*, KDChart::AbstractDiagram * *oldDiagram* = 0)

Replaces the old diagram, or appends the new diagram, it there is none yet.

Parameters:

newDiagram The diagram to be used instead of the old one. If this parameter is zero, the first diagram will just be removed.

oldDiagram The diagram to be removed by the new one. This diagram will be deleted automatically. If the parameter is omitted, the very first diagram will be replaced. In case, there was no diagram yet, the new diagram will just be added.

See also:

diagram, diagrams, addDiagram, removeDiagram, removeDiagrams, setDiagram

Definition at line 380 of file KDChartLegend.cpp.

References addDiagram(), d, and removeDiagram().

Referenced by setDiagram().

```
{
    KDChart::AbstractDiagram* old = oldDiagram;
    if( ! d->observers.isEmpty() && ! old ){
        old = d->observers.first()->diagram();
        if( ! old )
            d->observers.removeFirst(); // first entry had a 0 diagram
    }
    if( old )
        removeDiagram( old );
    if( newDiagram )
        addDiagram( newDiagram );
}
```

7.33.4.45 void Legend::resetDiagram (AbstractDiagram *)

Definition at line 400 of file KDChartLegend.cpp.

References removeDiagram().

Referenced by addDiagram().

```
{
    //qDebug() << oldDiagram;
    removeDiagram( oldDiagram );
}
```

7.33.4.46 void Legend::resetTexts ()

Removes all legend texts that might have been set by setText.

This resets the Legend to default behaviour: Texts are created automatically.

Definition at line 505 of file KDChartLegend.cpp.

References d, and setNeedRebuild().

```
{
    if( ! d->texts.count() ) return;
    d->texts.clear();
    setNeedRebuild();
}
```

7.33.4.47 void Legend::resizeEvent (QResizeEvent ∗ *event*) `[virtual]`

Definition at line 760 of file KDChartLegend.cpp.

References forceRebuild(), and sizeHint().

```
{
#ifdef DEBUG_LEGEND_PAINT
    qDebug() << "Legend::resizeEvent() called";
#endif
    forceRebuild();
    sizeHint();
    QTimer::singleShot(0, this, SLOT(emitPositionChanged()));
}
```

7.33.4.48 void Legend::resizeLayout (const QSize & *size*) `[virtual]`

Reimplemented from KDChart::AbstractAreaWidget.

Definition at line 168 of file KDChartLegend.cpp.

References activateTheLayout(), and d.

```
{
#ifdef DEBUG_LEGEND_PAINT
    qDebug() << "Legend::resizeLayout started";
#endif
    if( d->layout ){
        d->layout->setGeometry( QRect(QPoint(0,0), size) );
        activateTheLayout();
    }
#ifdef DEBUG_LEGEND_PAINT
    qDebug() << "Legend::resizeLayout done";
#endif
}
```

7.33.4.49 void Legend::setAlignment (Qt::Alignment)

Specify the alignment of a non-floating legend.

Use setFloatingPosition to set position and alignment if your legend is floating.

See also:

alignment, setPosition, setFloatingPosition

Definition at line 437 of file KDChartLegend.cpp.

References d.

Referenced by clone().

```
{
    d->alignment = alignment;
    emitPositionChanged();
}
```

7.33.4.50 void AbstractAreaBase::setBackgroundAttributes (const BackgroundAttributes & *a*) `[inherited]`

Definition at line 107 of file KDChartAbstractAreaBase.cpp.

References d.

```
{
    d->backgroundAttributes = a;
}
```

7.33.4.51 void Legend::setBrush (uint *dataset*, const QBrush & *brush*)

Definition at line 542 of file KDChartLegend.cpp.

References d, and setNeedRebuild().

```
{
    if( d->brushes[ dataset ] == brush ) return;
    d->brushes[ dataset ] = brush;
    setNeedRebuild();
}
```

7.33.4.52 void Legend::setBrushesFromDiagram (KDChart::AbstractDiagram ∗ *diagram*)

Definition at line 563 of file KDChartLegend.cpp.

References d, KDChart::AbstractDiagram::datasetBrushes(), and setNeedRebuild().

```
{
    bool bChangesDone = false;
    QList<QBrush> datasetBrushes = diagram->datasetBrushes();
    for( int i = 0; i < datasetBrushes.count(); i++ ){
        if( d->brushes[ i ] != datasetBrushes[ i ] ){
            d->brushes[ i ]  = datasetBrushes[ i ];
            bChangesDone = true;
        }
    }
    if( bChangesDone )
        setNeedRebuild();
}
```

7.33.4.53 void Legend::setColor (uint *dataset*, const QColor & *color*)

Note: there is no color() getter method, since setColor just sets a QBrush with the respective color, so the brush() getter method is sufficient.

Definition at line 535 of file KDChartLegend.cpp.

References d, and setNeedRebuild().

Referenced by setDefaultColors(), setRainbowColors(), and setSubduedColors().

```
536 {
537     if( d->brushes[ dataset ] == color ) return;
538     d->brushes[ dataset ] = color;
539     setNeedRebuild();
540 }
```

7.33.4.54 void Legend::setDefaultColors ()

Definition at line 682 of file KDChartLegend.cpp.

References setColor().

```
683 {
684     setColor(  0, Qt::red );
685     setColor(  1, Qt::green );
686     setColor(  2, Qt::blue );
687     setColor(  3, Qt::cyan );
688     setColor(  4, Qt::magenta );
689     setColor(  5, Qt::yellow );
690     setColor(  6, Qt::darkRed );
691     setColor(  7, Qt::darkGreen );
692     setColor(  8, Qt::darkBlue );
693     setColor(  9, Qt::darkCyan );
694     setColor( 10, Qt::darkMagenta );
695     setColor( 11, Qt::darkYellow );
696 }
```

7.33.4.55 void Legend::setDiagram (KDChart::AbstractDiagram $*$ *newDiagram*)

A convenience method doing the same as replaceDiagram(newDiagram, 0);.

Replaces the first diagram by the given diagram. If the legend's list of diagram is empty the given diagram is added to the list.

See also:
diagram, diagrams, addDiagram, removeDiagram, removeDiagrams, replaceDiagram

Definition at line 395 of file KDChartLegend.cpp.

References replaceDiagram().

Referenced by KDChart::Widget::addLegend(), Legend(), and KDChart::Widget::replaceLegend().

```
396 {
397     replaceDiagram( newDiagram );
398 }
```

7.33.4.56 void Legend::setFloatingPosition (const RelativePosition & *relativePosition*)

Specify the position and alignment of a floating legend.

Use setPosition and setAlignment to set position and alignment if your legend is non-floating.

Note:

When setFloatingPosition is called, the Legend's position value is set to KDChart::Position::Floating automatically.

If your Chart is pointed to by m_chart, your could have the floating legend alligned exactly to the chart's coordinate plane's top-right corner with the following commands:

```
KDChart::RelativePosition relativePosition;
relativePosition.setReferenceArea( m_chart->coordinatePlane() );
relativePosition.setReferencePosition( Position::NorthEast );
relativePosition.setAlignment( Qt::AlignTop | Qt::AlignRight );
relativePosition.setHorizontalPadding(
    KDChart::Measure( -1.0, KDChartEnums::MeasureCalculationModeAbsolute ) );
relativePosition.setVerticalPadding(
    KDChart::Measure( 0.0, KDChartEnums::MeasureCalculationModeAbsolute ) );
m_legend->setFloatingPosition( relativePosition );
```

To have the legend positioned at a fixed point, measured from the QPainter's top left corner, you could use the following code code:

```
KDChart::RelativePosition relativePosition;
relativePosition.setReferencePoints( PositionPoints( QPointF( 0.0, 0.0 ) ) );
relativePosition.setReferencePosition( Position::NorthWest );
relativePosition.setAlignment( Qt::AlignTop | Qt::AlignLeft );
relativePosition.setHorizontalPadding(
    KDChart::Measure( 4.0, KDChartEnums::MeasureCalculationModeAbsolute ) );
relativePosition.setVerticalPadding(
    KDChart::Measure( 4.0, KDChartEnums::MeasureCalculationModeAbsolute ) );
m_legend->setFloatingPosition( relativePosition );
```

Actually that's exactly the code KD Chart is using as default position for any floating legends, so if you just say setPosition(KDChart::Position::Floating) without calling setFloatingPosition your legend will be positioned at point 4/4.

See also:

setPosition, setAlignment

Definition at line 448 of file KDChartLegend.cpp.

References d.

```
{
    d->position = Position::Floating;
    if( d->relativePosition != relativePosition ){
        d->relativePosition  = relativePosition;
        emitPositionChanged();
    }
}
```

7.33.4.57 void AbstractAreaBase::setFrameAttributes (const FrameAttributes & *a*) `[inherited]`

Definition at line 97 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by clone().

```
98 {
99     d->frameAttributes = a;
100 }
```

7.33.4.58 void Legend::setLegendStyle (LegendStyle *style*)

Definition at line 189 of file KDChartLegend.cpp.

References d, and setNeedRebuild().

Referenced by clone().

```
190 {
191     if( d->legendStyle == style ) return;
192     d->legendStyle = style;
193     setNeedRebuild();
194 }
```

7.33.4.59 void Legend::setMarkerAttributes (uint *dataset*, const MarkerAttributes &)

Note that any sizes specified via setMarkerAttributes are ignored, unless you disable the automatic size calculation, by saying setUseAutomaticMarkerSize(false).

Definition at line 599 of file KDChartLegend.cpp.

References d, and setNeedRebuild().

```
600 {
601     if( d->markerAttributes[dataset] == markerAttributes ) return;
602     d->markerAttributes[ dataset ] = markerAttributes;
603     setNeedRebuild();
604 }
```

7.33.4.60 void Legend::setNeedRebuild ()

Definition at line 412 of file KDChartLegend.cpp.

References buildLegend(), and sizeHint().

Referenced by addDiagram(), removeDiagram(), resetTexts(), setBrush(), setBrushesFromDiagram(), setColor(), setLegendStyle(), setMarkerAttributes(), setOrientation(), setPen(), setReferenceArea(), setShowLines(), setSpacing(), setText(), setTextAttributes(), setTitleText(), setTitleTextAttributes(), and setUseAutomaticMarkerSize().

```
413 {
414     //qDebug() << "setNeedRebuild()";
415     buildLegend();
416     sizeHint();
417 }
```

7.33.4.61 void Legend::setOrientation (Qt::Orientation *orientation*)

Definition at line 462 of file KDChartLegend.cpp.

References d, and setNeedRebuild().

```
463 {
464     if( d->orientation == orientation ) return;
465     d->orientation = orientation;
466     setNeedRebuild();
467     emitPositionChanged();
468 }
```

7.33.4.62 void Legend::setPen (uint *dataset*, const QPen & *pen*)

Definition at line 578 of file KDChartLegend.cpp.

References d, and setNeedRebuild().

```
579 {
580     if( d->pens[dataset] == pen ) return;
581     d->pens[dataset] = pen;
582     setNeedRebuild();
583 }
```

7.33.4.63 void Legend::setPosition (Position *position*)

Specify the position of a non-floating legend.

Use setFloatingPosition to set position and alignment if your legend is floating.

See also:

setAlignment, setFloatingPosition

Definition at line 419 of file KDChartLegend.cpp.

References d.

Referenced by KDChart::Widget::addLegend(), and clone().

```
420 {
421     d->position = position;
422     emitPositionChanged();
423 }
```

7.33.4.64 void Legend::setRainbowColors ()

Definition at line 698 of file KDChartLegend.cpp.

References brush(), and setColor().

```
699 {
700     setColor(  0, QColor(255,  0,196) );
701     setColor(  1, QColor(255,  0, 96) );
702     setColor(  2, QColor(255, 128,64) );
```

```
    setColor(  3, Qt::yellow );
    setColor(  4, Qt::green );
    setColor(  5, Qt::cyan );
    setColor(  6, QColor( 96, 96,255) );
    setColor(  7, QColor(160,  0,255) );
    for( int i = 8; i < 16; ++i )
        setColor( i, brush( i - 8 ).color().light() );
}
```

7.33.4.65 void Legend::setReferenceArea (const QWidget ∗ *area*)

Specifies the reference area for font size of title text, and for font size of the item texts, IF automatic area detection is set.

Note:

This parameter is ignored, if the Measure given for setTitleTextAttributes (or setTextAttributes, resp.) is not specifying automatic area detection.

If no reference area is specified, but automatic area detection is set, then the size of the legend's parent widget will be used.

See also:

KDChart::Measure, KDChartEnums::MeasureCalculationMode

Definition at line 297 of file KDChartLegend.cpp.

References d, and setNeedRebuild().

Referenced by KDChart::Chart::addLegend().

```
{
    if( area == d->referenceArea ) return;
    d->referenceArea = area;
    setNeedRebuild();
}
```

7.33.4.66 void Legend::setShowLines (bool *legendShowLines*)

Definition at line 475 of file KDChartLegend.cpp.

References d, and setNeedRebuild().

```
{
    if( d->showLines == legendShowLines ) return;
    d->showLines = legendShowLines;
    setNeedRebuild();
    emitPositionChanged();
}
```

7.33.4.67 void Legend::setSpacing (uint *space*)

Definition at line 669 of file KDChartLegend.cpp.

References d, and setNeedRebuild().

```
{
    if( d->spacing == space && d->layout->spacing() == static_cast<int>(space) ) return;
    d->spacing = space;
    d->layout->setSpacing( space );
    setNeedRebuild();
}
```

7.33.4.68 void Legend::setSubduedColors (bool *ordered* = false)

Definition at line 712 of file KDChartLegend.cpp.

References setColor().

```
{
static const int NUM_SUBDUEDCOLORS = 18;
static const QColor SUBDUEDCOLORS[ NUM_SUBDUEDCOLORS ] = {
    QColor( 0xe0,0x7f,0x70 ),
    QColor( 0xe2,0xa5,0x6f ),
    QColor( 0xe0,0xc9,0x70 ),
    QColor( 0xd1,0xe0,0x70 ),
    QColor( 0xac,0xe0,0x70 ),
    QColor( 0x86,0xe0,0x70 ),
    QColor( 0x70,0xe0,0x7f ),
    QColor( 0x70,0xe0,0xa4 ),
    QColor( 0x70,0xe0,0xc9 ),
    QColor( 0x70,0xd1,0xe0 ),
    QColor( 0x70,0xac,0xe0 ),
    QColor( 0x70,0x86,0xe0 ),
    QColor( 0x7f,0x70,0xe0 ),
    QColor( 0xa4,0x70,0xe0 ),
    QColor( 0xc9,0x70,0xe0 ),
    QColor( 0xe0,0x70,0xd1 ),
    QColor( 0xe0,0x70,0xac ),
    QColor( 0xe0,0x70,0x86 ),
};
    if( ordered )
        for(int i=0; i<NUM_SUBDUEDCOLORS; ++i)
            setColor( i, SUBDUEDCOLORS[i] );
    else{
        setColor( 0, SUBDUEDCOLORS[ 0] );
        setColor( 1, SUBDUEDCOLORS[ 5] );
        setColor( 2, SUBDUEDCOLORS[10] );
        setColor( 3, SUBDUEDCOLORS[15] );
        setColor( 4, SUBDUEDCOLORS[ 2] );
        setColor( 5, SUBDUEDCOLORS[ 7] );
        setColor( 6, SUBDUEDCOLORS[12] );
        setColor( 7, SUBDUEDCOLORS[17] );
        setColor( 8, SUBDUEDCOLORS[ 4] );
        setColor( 9, SUBDUEDCOLORS[ 9] );
        setColor(10, SUBDUEDCOLORS[14] );
        setColor(11, SUBDUEDCOLORS[ 1] );
        setColor(12, SUBDUEDCOLORS[ 6] );
        setColor(13, SUBDUEDCOLORS[11] );
        setColor(14, SUBDUEDCOLORS[16] );
        setColor(15, SUBDUEDCOLORS[ 3] );
        setColor(16, SUBDUEDCOLORS[ 8] );
        setColor(17, SUBDUEDCOLORS[13] );
    }
}
```

7.33.4.69 void Legend::setText (uint *dataset*, const QString & *text*)

Definition at line 512 of file KDChartLegend.cpp.

References d, and setNeedRebuild().

```
{
    if( d->texts[ dataset ] == text ) return;
    d->texts[ dataset ] = text;
    setNeedRebuild();
}
```

7.33.4.70 void Legend::setTextAttributes (const TextAttributes & *a*)

Definition at line 621 of file KDChartLegend.cpp.

References d, and setNeedRebuild().

Referenced by KDChart::Chart::addLegend(), and clone().

```
{
    if( d->textAttributes == a ) return;
    d->textAttributes = a;
    setNeedRebuild();
}
```

7.33.4.71 void Legend::setTitleText (const QString & *text*)

Definition at line 633 of file KDChartLegend.cpp.

References d, and setNeedRebuild().

```
{
    if( d->titleText == text ) return;
    d->titleText = text;
    setNeedRebuild();
}
```

7.33.4.72 void Legend::setTitleTextAttributes (const TextAttributes & *a*)

Definition at line 645 of file KDChartLegend.cpp.

References d, and setNeedRebuild().

Referenced by KDChart::Chart::addLegend(), and clone().

```
{
    if( d->titleTextAttributes == a ) return;
    d->titleTextAttributes = a;
    setNeedRebuild();
}
```

7.33.4.73 void Legend::setUseAutomaticMarkerSize (bool *useAutomaticMarkerSize*)

This option is on by default, it means that Marker sizes in the Legend will be the same as the font height used for their respective label texts.

Set this to false, if you want to specify the marker sizes via setMarkerAttributes or if you want the Legend to use the same marker sizes as they are used in the Diagrams.

Definition at line 488 of file KDChartLegend.cpp.

References d, and setNeedRebuild().

Referenced by clone().

```
489 {
490     d->useAutomaticMarkerSize = useAutomaticMarkerSize;
491     setNeedRebuild();
492     emitPositionChanged();
493 }
```

7.33.4.74 void Legend::setVisible (bool *visible*) `[virtual]`

Definition at line 406 of file KDChartLegend.cpp.

Referenced by KDChart::Chart::addLegend().

```
407 {
408     QWidget::setVisible( visible );
409     emitPositionChanged();
410 }
```

7.33.4.75 bool Legend::showLines () const

Definition at line 483 of file KDChartLegend.cpp.

References d.

Referenced by buildLegend().

```
484 {
485     return d->showLines;
486 }
```

7.33.4.76 QSize Legend::sizeHint () const `[virtual]`

Definition at line 150 of file KDChartLegend.cpp.

References d.

Referenced by minimumSizeHint(), KDChart::Chart::reLayoutFloatingLegends(), resizeEvent(), and setNeedRebuild().

```
151 {
152 #ifdef DEBUG_LEGEND_PAINT
153     qDebug()  << "Legend::sizeHint() started";
154 #endif
155     Q_FOREACH( KDChart::AbstractLayoutItem* layoutItem, d->layoutItems ) {
156         layoutItem->sizeHint();
157     }
158     return AbstractAreaWidget::sizeHint();
159 }
```

7.33.4.77 uint Legend::spacing () const

Definition at line 677 of file KDChartLegend.cpp.

References d.

Referenced by buildLegend().

```
{
    return d->spacing;
}
```

7.33.4.78 QString Legend::text (uint *dataset*) const

Definition at line 519 of file KDChartLegend.cpp.

References d.

Referenced by buildLegend().

```
{
    if( d->texts.find( dataset ) != d->texts.end() ){
        //qDebug() << "Legend::text(" << dataset << ") returning d->texts[" << dataset << "] :" << d->
        return d->texts[ dataset ];
    }else{
        //qDebug() << "Legend::text(" << dataset << ") returning d->modelLabels[" << dataset << "] :"
        return d->modelLabels[ dataset ];
    }
}
```

7.33.4.79 TextAttributes Legend::textAttributes () const

Definition at line 628 of file KDChartLegend.cpp.

References d.

Referenced by KDChart::Chart::addLegend(), buildLegend(), and clone().

```
{
    return d->textAttributes;
}
```

7.33.4.80 const QMap< uint, QString > Legend::texts () const

Definition at line 530 of file KDChartLegend.cpp.

References d.

```
{
    return d->texts;
}
```

7.33.4.81 QString Legend::titleText () const

Definition at line 640 of file KDChartLegend.cpp.

References d.

Referenced by buildLegend().

```
641 {
642     return d->titleText;
643 }
```

7.33.4.82 TextAttributes Legend::titleTextAttributes () const

Definition at line 652 of file KDChartLegend.cpp.

References d.

Referenced by KDChart::Chart::addLegend(), buildLegend(), and clone().

```
653 {
654     return d->titleTextAttributes;
655 }
```

7.33.4.83 bool Legend::useAutomaticMarkerSize () const

Definition at line 495 of file KDChartLegend.cpp.

References d.

Referenced by buildLegend(), and clone().

```
496 {
497     return d->useAutomaticMarkerSize;
498 }
```

7.33.5 Member Data Documentation

7.33.5.1 Q_SIGNALS KDChart::Legend::__pad0__

Reimplemented from KDChart::AbstractAreaWidget.

Definition at line 354 of file KDChartLegend.h.

7.33.5.2 private KDChart::Legend::Q_SLOTS

Definition at line 359 of file KDChartLegend.h.

The documentation for this class was generated from the following files:

- KDChartLegend.h
- KDChartLegend.cpp

7.34 KDChart::LineAttributes Class Reference

`#include <KDChartLineAttributes.h>`

Public Types

- enum MissingValuesPolicy {
  MissingValuesAreBridged,
  MissingValuesHideSegments,
  MissingValuesShownAsZero,
  MissingValuesPolicyIgnored }

  MissingValuesPolicy specifies how a missing value will be shown in a line diagram.

Public Member Functions

- bool displayArea () const
- LineAttributes (const LineAttributes &)
- LineAttributes ()
- MissingValuesPolicy missingValuesPolicy () const
- bool operator!= (const LineAttributes &other) const
- LineAttributes & operator= (const LineAttributes &)
- bool operator== (const LineAttributes &) const
- void setDisplayArea (bool display)
- void setMissingValuesPolicy (MissingValuesPolicy policy)
- void setTransparency (uint alpha)
- uint transparency () const
- ~LineAttributes ()

7.34.1 Member Enumeration Documentation

7.34.1.1 enum KDChart::LineAttributes::MissingValuesPolicy

MissingValuesPolicy specifies how a missing value will be shown in a line diagram.

Missing value is assumed if the data cell contains a QVariant that can not be interpreted as a double, or if the data cell is hidden while its dataset is not hidden.

- `MissingValuesAreBridged` the default: No markers will be shown for missing values but the line will be bridged if there is at least one valid cell before and after the missing value(s), otherwise the segment will be hidden.

- `MissingValuesHideSegments` Line segments starting with a missing value will not be shown, and no markers will be shown for missing values, so this will look like a piece of the line is missing.

- `MissingValuesShownAsZero` Missing value(s) will be treated like normal zero values, and markers will shown for them too, so there will be no visible difference between a zero value and a missing value.

- `MissingValuesPolicyIgnored` (internal value, do not use)

Enumeration values:

MissingValuesAreBridged

MissingValuesHideSegments

MissingValuesShownAsZero

MissingValuesPolicyIgnored

Definition at line 55 of file KDChartLineAttributes.h.

```
55                              {
56         MissingValuesAreBridged,
57         MissingValuesHideSegments,
58         MissingValuesShownAsZero,
59         MissingValuesPolicyIgnored };
```

7.34.2 Constructor & Destructor Documentation

7.34.2.1 KDChart::LineAttributes::LineAttributes ()

7.34.2.2 KDChart::LineAttributes::LineAttributes (const LineAttributes &)

7.34.2.3 KDChart::LineAttributes::~LineAttributes ()

7.34.3 Member Function Documentation

7.34.3.1 bool KDChart::LineAttributes::displayArea () const

Referenced by operator<<(), and KDChart::LineDiagram::paint().

7.34.3.2 MissingValuesPolicy KDChart::LineAttributes::missingValuesPolicy () const

Referenced by KDChart::LineDiagram::getCellValues().

7.34.3.3 bool KDChart::LineAttributes::operator!= (const LineAttributes & *other*) const

Definition at line 79 of file KDChartLineAttributes.h.

```
79 { return !operator==(other); }
```

7.34.3.4 LineAttributes& KDChart::LineAttributes::operator= (const LineAttributes &)

7.34.3.5 bool KDChart::LineAttributes::operator== (const LineAttributes &) const

7.34.3.6 void KDChart::LineAttributes::setDisplayArea (bool *display*)

7.34.3.7 void KDChart::LineAttributes::setMissingValuesPolicy (MissingValuesPolicy *policy*)

7.34.3.8 void KDChart::LineAttributes::setTransparency (uint *alpha*)

7.34.3.9 uint KDChart::LineAttributes::transparency () const

Referenced by operator<<(), and KDChart::LineDiagram::paint().

The documentation for this class was generated from the following file:

- KDChartLineAttributes.h

7.35 KDChart::LineDiagram Class Reference

`#include <KDChartLineDiagram.h>`

Inheritance diagram for KDChart::LineDiagram:Collaboration diagram for KDChart::LineDiagram:

Public Types

- enum LineType {
  Normal = 0,
  Stacked = 1,
  Percent = 2 }

Public Member Functions

- virtual void addAxis (CartesianAxis ∗axis)

  Add the axis to the diagram.

- bool allowOverlappingDataValueTexts () const
- bool antiAliasing () const
- virtual AttributesModel ∗ attributesModel () const

  Returns the AttributesModel, that is used by this diagram.

- virtual KDChart::CartesianAxisList axes () const
- QBrush brush (const QModelIndex &index) const

  Retrieve the brush to be used, for painting the datapoint at the given index in the model.

- QBrush brush (int dataset) const

  Retrieve the brush to be used for the given dataset.

- QBrush brush () const

  Retrieve the brush to be used for painting datapoints globally.

- virtual LineDiagram ∗ clone () const
- bool compare (const AbstractDiagram ∗other) const

  Returns true if both diagrams have the same settings.

- bool compare (const AbstractCartesianDiagram ∗other) const

  Returns true if both diagrams have the same settings.

- bool compare (const LineDiagram ∗other) const

  Returns true if both diagrams have the same settings.

- AbstractCoordinatePlane ∗ coordinatePlane () const

  The coordinate plane associated with the diagram.

- const QPair< QPointF, QPointF > dataBoundaries () const

  Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

- virtual void dataChanged (const QModelIndex &topLeft, const QModelIndex &bottomRight)

  [reimplemented]

- QList< QBrush > datasetBrushes () const

  The set of dataset brushes currently used, for use in legends, etc.

- int datasetDimension () const

  The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

- QStringList datasetLabels () const

  The set of dataset labels currently displayed, for use in legends, etc.

- QList< MarkerAttributes > datasetMarkers () const

  The set of dataset markers currently used, for use in legends, etc.

- QList< QPen > datasetPens () const

  The set of dataset pens currently used, for use in legends, etc.

- DataValueAttributes dataValueAttributes (const QModelIndex &index) const

  Retrieve the DataValueAttributes for the given index.

- DataValueAttributes dataValueAttributes (int column) const

  Retrieve the DataValueAttributes for the given dataset.

- DataValueAttributes dataValueAttributes () const

  Retrieve the DataValueAttributes speficied globally.

- virtual void doItemsLayout ()

  [reimplemented]

- virtual int horizontalOffset () const

  [reimplemented]

- virtual QModelIndex indexAt (const QPoint &point) const

  [reimplemented]

- bool isHidden (const QModelIndex &index) const

  Retrieve the hidden status for the given index.

- bool isHidden (int column) const

  Retrieve the hidden status for the given dataset.

- bool isHidden () const

  Retrieve the hidden status speficied globally.

- virtual bool isIndexHidden (const QModelIndex &index) const

  [reimplemented]

- QStringList itemRowLabels () const

  The set of item row labels currently displayed, for use in Abscissa axes, etc.

- virtual void layoutPlanes ()
- LineAttributes lineAttributes (const QModelIndex &index) const
- LineAttributes lineAttributes (int column) const
- LineAttributes lineAttributes () const
- LineDiagram (QWidget ∗parent=0, CartesianCoordinatePlane ∗plane=0)
- virtual QModelIndex moveCursor (CursorAction cursorAction, Qt::KeyboardModifiers modifiers)

  [reimplemented]

- const int numberOfAbscissaSegments () const
- const int numberOfOrdinateSegments () const
- void paintDataValueText (QPainter ∗painter, const QModelIndex &index, const QPointF &pos, double value)
- void paintMarker (QPainter ∗painter, const QModelIndex &index, const QPointF &pos)
- virtual void paintMarker (QPainter ∗painter, const MarkerAttributes &markerAttributes, const QBrush &brush, const QPen &, const QPointF &point, const QSizeF &size)
- QPen pen (const QModelIndex &index) const

  Retrieve the pen to be used, for painting the datapoint at the given index in the model.

- QPen pen (int dataset) const

  Retrieve the pen to be used for the given dataset.

- QPen pen () const

  Retrieve the pen to be used for painting datapoints globally.

- bool percentMode () const
- virtual AbstractCartesianDiagram ∗ referenceDiagram () const
- virtual QPointF referenceDiagramOffset () const
- void resetLineAttributes (const QModelIndex &index)

  Remove any explicit line attributes settings that might have been specified before.

- void resetLineAttributes (int column)
- void resize (const QSizeF &area)

  Called by the widget's sizeEvent.

- virtual void scrollTo (const QModelIndex &index, ScrollHint hint=EnsureVisible)

  [reimplemented]

- void setAllowOverlappingDataValueTexts (bool allow)

  Set whether data value labels are allowed to overlap.

- void setAntiAliasing (bool enabled)

  Set whether anti-aliasing is to be used while rendering this diagram.

- virtual void setAttributesModel (AttributesModel ∗model)

  Associate an AttributesModel with this diagram.

- void setBrush (const QBrush &brush)

Set the brush to be used, for painting all datasets in the model.

- void setBrush (int dataset, const QBrush &brush)

  Set the brush to be used, for painting the given dataset.

- void setBrush (const QModelIndex &index, const QBrush &brush)

  Set the brush to be used, for painting the datapoint at the given index.

- virtual void setCoordinatePlane (AbstractCoordinatePlane ∗plane)

  Set the coordinate plane associated with the diagram.

- void setDatasetDimension (int dimension)

  Sets the dataset dimension of the diagram.

- void setDataValueAttributes (const DataValueAttributes &a)

  Set the DataValueAttributes for all datapoints in the model.

- void setDataValueAttributes (int dataset, const DataValueAttributes &a)

  Set the DataValueAttributes for the given dataset.

- void setDataValueAttributes (const QModelIndex &index, const DataValueAttributes &a)

  Set the DataValueAttributes for the given index.

- void setHidden (bool hidden)

  Hide (or unhide, resp.) all datapoints in the model.

- void setHidden (int column, bool hidden)

  Hide (or unhide, resp.) a dataset.

- void setHidden (const QModelIndex &index, bool hidden)

  Hide (or unhide, resp.) a data cell.

- void setLineAttributes (const QModelIndex &index, const LineAttributes &a)
- void setLineAttributes (int column, const LineAttributes &a)
- void setLineAttributes (const LineAttributes &a)
- virtual void setModel (QAbstractItemModel ∗model)

  Associate a model with the diagram.

- void setPen (const QPen &pen)

  Set the pen to be used, for painting all datasets in the model.

- void setPen (int dataset, const QPen &pen)

  Set the pen to be used, for painting the given dataset.

- void setPen (const QModelIndex &index, const QPen &pen)

  Set the pen to be used, for painting the datapoint at the given index.

- void setPercentMode (bool percent)
- virtual void setReferenceDiagram (AbstractCartesianDiagram ∗diagram, const QPointF &offset=QPointF())

- virtual void setRootIndex (const QModelIndex &idx)

  Set the root index in the model, where the diagram starts referencing data for display.

- virtual void setSelection (const QRect &rect, QItemSelectionModel::SelectionFlags command)

  [reimplemented]

- void setThreeDLineAttributes (const QModelIndex &index, const ThreeDLineAttributes &a)
- void setThreeDLineAttributes (int column, const ThreeDLineAttributes &a)
- void setThreeDLineAttributes (const ThreeDLineAttributes &a)
- void setType (const LineType type)
- virtual void takeAxis (CartesianAxis $*$axis)

  Removes the axis from the diagram, without deleting it.

- ThreeDLineAttributes threeDLineAttributes (const QModelIndex &index) const
- ThreeDLineAttributes threeDLineAttributes (int column) const
- ThreeDLineAttributes threeDLineAttributes () const
- LineType type () const
- void update () const
- void useDefaultColors ()

  Set the palette to be used, for painting datasets to the default palette.

- void useRainbowColors ()

  Set the palette to be used, for painting datasets to the rainbow palette.

- virtual bool usesExternalAttributesModel () const

  Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

- void useSubduedColors ()

  Set the palette to be used, for painting datasets to the subdued palette.

- virtual int verticalOffset () const

  [reimplemented]

- virtual QRect visualRect (const QModelIndex &index) const

  [reimplemented]

- virtual QRegion visualRegionForSelection (const QItemSelection &selection) const

  [reimplemented]

- virtual ~LineDiagram ()

Protected Member Functions

- QModelIndex attributesModelRootIndex () const
- virtual const QPair< QPointF, QPointF > calculateDataBoundaries () const

  [reimplemented]

- virtual bool checkInvariants (bool justReturnTheStatus=false) const
- QModelIndex columnToIndex (int column) const

- void dataHidden ()

  This signal is emitted, when the hidden status of at least one data cell was (un)set.

- LineAttributes::MissingValuesPolicy getCellValues (int row, int column, bool shiftCountedXValuesByHalfSection, double &valueX, double &valueY) const
- void modelsChanged ()

  This signal is emitted, when either the model or the AttributesModel is replaced.

- void paint (PaintContext ∗paintContext)

  Draw the diagram contents to the rectangle and painter, that are passed in as part of the paint context.

- virtual void paintDataValueTexts (QPainter ∗painter)
- void paintEvent (QPaintEvent ∗)
- virtual void paintMarkers (QPainter ∗painter)
- void propertiesChanged ()

  Emitted upon change of a property of the Diagram.

- void resizeEvent (QResizeEvent ∗)
- void setAttributesModelRootIndex (const QModelIndex &)
- void setDataBoundariesDirty () const
- virtual double threeDItemDepth (int column) const
- virtual double threeDItemDepth (const QModelIndex &index) const
- double valueForCell (int row, int column) const

  Helper method, retrieving the data value (DisplayRole) for a given row and column.

- double valueForCellTesting (int row, int column, bool &bOK, bool showHiddenCellsAsInvalid=false) const

Protected Attributes

- Q_SIGNALS __pad0__: void layoutChanged(AbstractDiagram∗)

7.35.1 Member Enumeration Documentation

7.35.1.1 enum KDChart::LineDiagram::LineType

Enumeration values:

Normal

Stacked

Percent

Definition at line 62 of file KDChartLineDiagram.h.

```
62                   {
63         Normal =  0,
64         Stacked = 1,
65         Percent = 2
66     };
```

7.35.2 Constructor & Destructor Documentation

7.35.2.1 LineDiagram::LineDiagram (QWidget ∗ *parent* = 0, CartesianCoordinatePlane ∗ *plane* = 0)

Definition at line 60 of file KDChartLineDiagram.cpp.

Referenced by clone().

```
60                                                                           :
61     AbstractCartesianDiagram( new Private(), parent, plane )
62 {
63     init();
64 }
```

7.35.2.2 LineDiagram::∼LineDiagram () `[virtual]`

Definition at line 70 of file KDChartLineDiagram.cpp.

```
71 {
72 }
```

7.35.3 Member Function Documentation

7.35.3.1 void AbstractCartesianDiagram::addAxis (CartesianAxis ∗ *axis*) `[virtual, inherited]`

Add the axis to the diagram.

The diagram takes ownership of the axis and will delete it.

To gain back ownership (e.g. for assigning the axis to another diagram) use the takeAxis method, before calling addAxis on the other diagram.

See also:

takeAxis

Definition at line 89 of file KDChartAbstractCartesianDiagram.cpp.

References KDChart::AbstractAxis::createObserver(), d, and KDChart::AbstractCartesianDiagram::layoutPlanes().

```
90 {
91     if ( !d->axesList.contains( axis ) ) {
92         d->axesList.append( axis );
93         axis->createObserver( this );
94         layoutPlanes();
95     }
96 }
```

7.35.3.2 bool AbstractDiagram::allowOverlappingDataValueTexts () const `[inherited]`

Returns:

Whether data value labels are allowed to overlap.

Definition at line 446 of file KDChartAbstractDiagram.cpp.

References d.

```
450 {
```

7.35.3.3 bool AbstractDiagram::antiAliasing () const `[inherited]`

Returns:

Whether anti-aliasing is to be used for rendering this diagram.

Definition at line 457 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by paint().

```
461 {
```

7.35.3.4 AttributesModel ∗ AbstractDiagram::attributesModel () const `[virtual, inherited]`

Returns the AttributesModel, that is used by this diagram.

By default each diagram owns its own AttributesModel, which should never be deleted. Only if a user-supplied AttributesModel has been set does the pointer returned here not belong to the diagram.

Returns:

The AttributesModel associated with the diagram.

See also:

setAttributesModel

Definition at line 286 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::RingDiagram::paint(), KDChart::PolarDiagram::paint(), and KDChart::BarDiagram::setBarAttributes().

```
287 {
288     return d->attributesModel;
289 }
```

7.35.3.5 QModelIndex AbstractDiagram::attributesModelRootIndex () const `[protected, inherited]`

returns a QModelIndex pointing into the AttributesModel that corresponds to the root index of the diagram.

Definition at line 310 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by calculateDataBoundaries(), KDChart::BarDiagram::calculateDataBoundaries(), numberOfAbscissaSegments(), KDChart::BarDiagram::numberOfAbscissaSegments(), numberOfOrdinateSegments(), KDChart::BarDiagram::numberOfOrdinateSegments(), paint(), KDChart::BarDiagram::paint(), and KDChart::AbstractDiagram::valueForCell().

```
316 {
```

7.35.3.6 KDChart::CartesianAxisList AbstractCartesianDiagram::axes () const `[virtual, inherited]`

Definition at line 108 of file KDChartAbstractCartesianDiagram.cpp.

References KDChart::CartesianAxisList, and d.

```
109 {
110     return d->axesList;
111 }
```

7.35.3.7 QBrush AbstractDiagram::brush (const QModelIndex & *index*) const `[inherited]`

Retrieve the brush to be used, for painting the datapoint at the given index in the model.

Parameters:

index The index of the datapoint in the model.

Returns:

The brush to use for painting.

Definition at line 816 of file KDChartAbstractDiagram.cpp.

```
822                                 :
QRect AbstractDiagram::visualRect(const QModelIndex &) const
```

7.35.3.8 QBrush AbstractDiagram::brush (int *dataset*) const `[inherited]`

Retrieve the brush to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the brush for.

Returns:

The brush to use for painting.

Definition at line 808 of file KDChartAbstractDiagram.cpp.

```
815 {
```

7.35.3.9 QBrush AbstractDiagram::brush () const `[inherited]`

Retrieve the brush to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:
The brush to use for painting.

Definition at line 802 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::PieDiagram::paint(), paint(), and KDChart::AbstractDiagram::paintMarker().

```
{
```

7.35.3.10 const QPair< QPointF, QPointF > LineDiagram::calculateDataBoundaries () const `[protected, virtual]`

[reimplemented]

Implements KDChart::AbstractDiagram.

Definition at line 267 of file KDChartLineDiagram.cpp.

References KDChart::AbstractDiagram::attributesModelRootIndex(), KDChart::AbstractDiagram::checkInvariants(), d, KDChart::AbstractDiagram::datasetDimension(), type(), and valueForCellTesting().

```
{
    if ( !checkInvariants( true ) ) return QPair<QPointF, QPointF>( QPointF( 0, 0 ), QPointF( 0, 0 ) )

    // note: calculateDataBoundaries() is ignoring the hidden flags.
    //       That’s not a bug but a feature: Hiding data does not mean removing them.
    // For totally removing data from KD Chart’s view people can use e.g. a proxy model ...

    const int rowCount = d->attributesModel->rowCount(attributesModelRootIndex());
    const int colCount = d->attributesModel->columnCount(attributesModelRootIndex());
    double xMin = 0;
    double xMax = rowCount -1;
    double yMin = 0, yMax = 0;
    bool bOK;

    // calculate boundaries for different line types Normal - Stacked - Percent - Default Normal
    switch ( type() ){
    case LineDiagram::Normal:
    {
        bool bStarting = true;
        for( int i = datasetDimension()-1; i < colCount; i += datasetDimension() ) {
            for ( int j=0; j< rowCount; ++j ) {
                const double value = valueForCellTesting( j, i, bOK );
                double xvalue;
                if( datasetDimension() > 1 && bOK )
                    xvalue = valueForCellTesting( j, i-1, bOK );
                if( bOK ){
                    if( bStarting ){
                        yMin = value;
                        yMax = value;
                    }else{
                        yMin = qMin( yMin, value );
                        yMax = qMax( yMax, value );
                    }
                    if ( datasetDimension() > 1 ) {
                        if( bStarting ){
                            xMin = xvalue;
                            xMax = xvalue;
                        }else{
                            xMin = qMin( xMin, xvalue );
                            xMax = qMax( xMax, xvalue );
                        }
                    }
                    bStarting = false;
                }
            }
        }

        // the following code is replaced by CartesianCoordinatePlane’s automatic range / zoom adjusti
        //if( yMin > 0 && yMax / yMin >= 2.0 )
        //    yMin = 0;
        //else if( yMax < 0 && yMax / yMin <= 0.5 )
        //    yMax = 0;
    }
    break;
    case LineDiagram::Stacked:
    {
        bool bStarting = true;
        for ( int j=0; j< rowCount; ++j ) {
            // calculate sum of values per column - Find out stacked Min/Max
            double stackedValues = 0;
            for( int i = datasetDimension()-1; i < colCount; i += datasetDimension() ) {
                const double value = valueForCellTesting( j, i, bOK );
                if( bOK )
                    stackedValues += value;
            }
            if( bStarting ){
                yMin = stackedValues;
                yMax = stackedValues;
                bStarting = false;
            }else{
                // Pending Michel:
                // I am taking in account all values negatives or positives
                // take in account all stacked values
                yMin = qMin( qMin( yMin,  0.0 ), stackedValues );
                yMax = qMax( yMax, stackedValues );
            }
        }
    }
    break;
    case LineDiagram::Percent:
    {
        for( int i = datasetDimension()-1; i < colCount; i += datasetDimension() ) {
            for ( int j=0; j< rowCount; ++j ) {
                // Ordinate should begin at 0 the max value being the 100% pos
                const double value = valueForCellTesting( j, i, bOK );
                if( bOK )
                    yMax = qMax( yMax, value );
            }
        }
    }
    break;
    default:
        Q_ASSERT_X ( false, "calculateDataBoundaries()",
                     "Type item does not match a defined line chart Type." );
    }

    QPointF bottomLeft( QPointF( xMin, yMin ) );
    QPointF topRight(   QPointF( xMax, yMax ) );
    //qDebug() << "LineDiagram::calculateDataBoundaries () returns ( " << bottomLeft << topRight <<")"
    return QPair<QPointF, QPointF> ( bottomLeft, topRight );
}
```

7.35.3.11 bool AbstractDiagram::checkInvariants (bool *justReturnTheStatus* = false) const `[protected, virtual, inherited]`

Definition at line 930 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane().

Referenced by KDChart::RingDiagram::calculateDataBoundaries(), KDChart::PolarDiagram::calculateDataBoundaries(), KDChart::PieDiagram::calculateDataBoundaries(), calculateDataBoundaries(), KDChart::BarDiagram::calculateDataBoundaries(), KDChart::RingDiagram::paint(), KDChart::PolarDiagram::paint(), KDChart::PieDiagram::paint(), paint(), KDChart::BarDiagram::paint(), and KDChart::AbstractDiagram::paintMarker().

```
930                        {
931         Q_ASSERT_X ( model(), "AbstractDiagram::checkInvariants()",
932                      "There is no usable model set, for the diagram." );
933
934         Q_ASSERT_X ( coordinatePlane(), "AbstractDiagram::checkInvariants()",
935                      "There is no usable coordinate plane set, for the diagram." );
936     }
937     return model() && coordinatePlane();
938 }
939
940 int AbstractDiagram::datasetDimension( ) const
```

7.35.3.12 LineDiagram ∗ LineDiagram::clone () const `[virtual]`

Definition at line 74 of file KDChartLineDiagram.cpp.

References d, and LineDiagram().

```
75 {
76     return new LineDiagram( new Private( *d ) );
77 }
```

7.35.3.13 QModelIndex AbstractDiagram::columnToIndex (int *column*) const `[protected, inherited]`

Definition at line 317 of file KDChartAbstractDiagram.cpp.

```
323 {
```

7.35.3.14 bool AbstractDiagram::compare (const AbstractDiagram ∗ *other*) const `[inherited]`

Returns true if both diagrams have the same settings.

Definition at line 135 of file KDChartAbstractDiagram.cpp.

```
136 {
137     if( other == this ) return true;
138     if( ! other ){
139         //qDebug() << "AbstractDiagram::compare() cannot compare to Null pointer";
140         return false;
141     }
```

```
    /*
    qDebug() << "\n            AbstractDiagram::compare() QAbstractScrollArea:";
            // compare QAbstractScrollArea properties
    qDebug() <<
            ((horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
            (verticalScrollBarPolicy()    == other->verticalScrollBarPolicy()));
    qDebug() << "AbstractDiagram::compare() QFrame:";
            // compare QFrame properties
    qDebug() <<
            ((frameShadow() == other->frameShadow()) &&
            (frameShape()   == other->frameShape()) &&
            (frameWidth()   == other->frameWidth()) &&
            (lineWidth()    == other->lineWidth()) &&
            (midLineWidth() == other->midLineWidth()));
    qDebug() << "AbstractDiagram::compare() QAbstractItemView:";
            // compare QAbstractItemView properties
    qDebug() <<
            ((alternatingRowColors() == other->alternatingRowColors()) &&
            (hasAutoScroll()         == other->hasAutoScroll()) &&
#if QT_VERSION > 0x040199
            (dragDropMode()          == other->dragDropMode()) &&
            (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
            (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
            (verticalScrollMode()    == other->verticalScrollMode()) &&
#endif
            (dragEnabled()           == other->dragEnabled()) &&
            (editTriggers()          == other->editTriggers()) &&
            (iconSize()              == other->iconSize()) &&
            (selectionBehavior()     == other->selectionBehavior()) &&
            (selectionMode()         == other->selectionMode()) &&
            (showDropIndicator()     == other->showDropIndicator()) &&
            (tabKeyNavigation()      == other->tabKeyNavigation()) &&
            (textElideMode()         == other->textElideMode()));
    qDebug() << "AbstractDiagram::compare() AttributesModel: ";
            // compare all of the properties stored in the attributes model
    qDebug() << attributesModel()->compare( other->attributesModel() );
    qDebug() << "AbstractDiagram::compare() own:";
            // compare own properties
    qDebug() <<
            ((rootIndex().column()            == other->rootIndex().column()) &&
            (rootIndex().row()                == other->rootIndex().row()) &&
            (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
            (antiAliasing()                   == other->antiAliasing()) &&
            (percentMode()                    == other->percentMode()) &&
            (datasetDimension()               == other->datasetDimension()));
    */
    return  // compare QAbstractScrollArea properties
            (horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
            (verticalScrollBarPolicy()   == other->verticalScrollBarPolicy()) &&
            // compare QFrame properties
            (frameShadow()  == other->frameShadow()) &&
            (frameShape()   == other->frameShape()) &&
            (frameWidth()   == other->frameWidth()) &&
            (lineWidth()    == other->lineWidth()) &&
            (midLineWidth() == other->midLineWidth()) &&
            // compare QAbstractItemView properties
            (alternatingRowColors()  == other->alternatingRowColors()) &&
            (hasAutoScroll()         == other->hasAutoScroll()) &&
#if QT_VERSION > 0x040199
            (dragDropMode()          == other->dragDropMode()) &&
            (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
            (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
            (verticalScrollMode()    == other->verticalScrollMode()) &&
#endif
            (dragEnabled()           == other->dragEnabled()) &&
            (editTriggers()          == other->editTriggers()) &&
            (iconSize()              == other->iconSize()) &&
209             (selectionBehavior()     == other->selectionBehavior()) &&
210             (selectionMode()         == other->selectionMode()) &&
211             (showDropIndicator()     == other->showDropIndicator()) &&
212             (tabKeyNavigation()      == other->tabKeyNavigation()) &&
213             (textElideMode()         == other->textElideMode()) &&
214             // compare all of the properties stored in the attributes model
215             attributesModel()->compare( other->attributesModel() ) &&
216             // compare own properties
217             (rootIndex().column()             == other->rootIndex().column()) &&
218             (rootIndex().row()                == other->rootIndex().row()) &&
219             (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
220             (antiAliasing()                   == other->antiAliasing()) &&
221             (percentMode()                    == other->percentMode()) &&
222             (datasetDimension()               == other->datasetDimension());
223 }
```

7.35.3.15 bool AbstractCartesianDiagram::compare (const AbstractCartesianDiagram ∗ *other*) const [inherited]

Returns true if both diagrams have the same settings.

Definition at line 52 of file KDChartAbstractCartesianDiagram.cpp.

```
53 {
54     if( other == this ) return true;
55     if( ! other ){
56         //qDebug() << "AbstractCartesianDiagram::compare() cannot compare to Null pointer";
57         return false;
58     }
59     /*
60     qDebug() << "\n             AbstractCartesianDiagram::compare():";
61             // compare own properties
62     qDebug() <<
63             ((referenceDiagram() == other->referenceDiagram()) &&
64             ((! referenceDiagram()) || (referenceDiagramOffset() == other->referenceDiagramOffset()))));
65     */
66     return  // compare the base class
67             ( static_cast<const AbstractDiagram*>(this)->compare( other ) ) &&
68             // compare own properties
69             (referenceDiagram() == other->referenceDiagram()) &&
70             ((! referenceDiagram()) || (referenceDiagramOffset() == other->referenceDiagramOffset()));
71 }
```

7.35.3.16 bool LineDiagram::compare (const LineDiagram ∗ *other*) const

Returns true if both diagrams have the same settings.

Definition at line 80 of file KDChartLineDiagram.cpp.

```
81 {
82     if( other == this ) return true;
83     if( ! other ){
84         //qDebug() << "LineDiagram::compare() cannot compare to Null pointer";
85         return false;
86     }
87     /*
88     qDebug() <<"\n             LineDiagram::compare():";
89             // compare own properties
90     qDebug() << (type() == other->type());
91     */
92     return  // compare the base class
93              ( static_cast<const AbstractCartesianDiagram*>(this)->compare( other ) ) &&
94              // compare own properties
95              (type() == other->type());
96 }
```

7.35.3.17 AbstractCoordinatePlane * AbstractDiagram::coordinatePlane () const `[inherited]`

The coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. By default this is a CartesianCoordinatePlane.

Returns:

The coordinate plane associated with the diagram.

Definition at line 226 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractCartesianDiagram::layoutPlanes(), KDChart::PolarDiagram::paint(), paint(), KDChart::BarDiagram::paint(), KDChart::AbstractPolarDiagram::polarCoordinatePlane(), and KDChart::AbstractCartesianDiagram::setCoordinatePlane().

```
227 {
228     return d->plane;
229 }
```

7.35.3.18 const QPair< QPointF, QPointF > AbstractDiagram::dataBoundaries () const `[inherited]`

Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

This method returns a chached result of calculations done by calculateDataBoundaries. Classes derived from AbstractDiagram must implement the calculateDataBoundaries function, to specify their own way of calculating the data boundaries. If derived classes want to force recalculation of the data boundaries, they can call setDataBoundariesDirty()

Returned value is in diagram coordinates.

Definition at line 231 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::calculateDataBoundaries(), and d.

Referenced by KDChart::CartesianCoordinatePlane::getRawDataBoundingRectFromDiagrams(), KDChart::PolarCoordinatePlane::layoutDiagrams(), paint(), and KDChart::BarDiagram::paint().

```
232 {
233     if( d->databoundariesDirty ){
234         d->databoundaries = calculateDataBoundaries ();
235         d->databoundariesDirty = false;
236     }
237     return d->databoundaries;
238 }
```

7.35.3.19 void AbstractDiagram::dataChanged (const QModelIndex & *topLeft*, const QModelIndex & *bottomRight*) `[virtual, inherited]`

[reimplemented]

Definition at line 338 of file KDChartAbstractDiagram.cpp.

References d.

```
338 {
339   // We are still too dumb to do intelligent updates...
340   d->databoundariesDirty = true;
341   scheduleDelayedItemsLayout();
342 }
343
344
```

7.35.3.20 void KDChart::AbstractDiagram::dataHidden () `[protected, inherited]`

This signal is emitted, when the hidden status of at least one data cell was (un)set.

7.35.3.21 QList< QBrush > AbstractDiagram::datasetBrushes () const `[inherited]`

The set of dataset brushes currently used, for use in legends, etc.

Note:

Cell-level override brushes, if set, take precedence over the dataset values, so you might need to check these too, in order to find the brush, that is used for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 894 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), KDChart::Legend::datasetCount(), and KDChart::Legend::setBrushesFromDiagram().

```
896                                                                                 {
897         QBrush brush = qVariantValue<QBrush>( attributesModel()->headerData( i, Qt::Vertical, DatasetB
898         ret << brush;
899     }
900
901     return ret;
902 }
903
904 QList<QPen> AbstractDiagram::datasetPens() const
```

7.35.3.22 int AbstractDiagram::datasetDimension () const `[inherited]`

The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

For each dimension it will expect one column of values in the model. If the dimensionality is 1, automatic values will be used for the abscissa.

For example a diagram with the default dimension of 1, will have one column per datapoint (the y values) and will use automatic values for the x axis (1, 2, 3, ... n). If the dimension is 2, the diagram will use the first, (and the third, fifth, etc) columns as X values, and the second, (and the fourth, sixth, etc) column as Y values.

Returns:

The dataset dimension of the diagram.

Definition at line 942 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by calculateDataBoundaries(), getCellValues(), KDChart::CartesianCoordinatePlane::getDataDimensionsList(), paint(), and setType().

```
946 {
```

7.35.3.23 QStringList AbstractDiagram::datasetLabels () const `[inherited]`

The set of dataset labels currently displayed, for use in legends, etc.

Returns:

The set of dataset labels currently displayed.

Definition at line 882 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), and KDChart::Legend::datasetCount().

```
883                                                   : " << attributesModel()->columnCount(attributesModel
884     const int columnCount = attributesModel()->columnCount(attributesModelRootIndex());
885     for( int i = datasetDimension()-1; i < columnCount; i += datasetDimension() ){
886         //qDebug() << "dataset label: " << attributesModel()->headerData( i, Qt::Horizontal, Qt::Displ
887         ret << attributesModel()->headerData( i, Qt::Horizontal, Qt::DisplayRole ).toString();
888     }
889     return ret;
890 }
891
892 QList<QBrush> AbstractDiagram::datasetBrushes() const
```

7.35.3.24 QList< MarkerAttributes > AbstractDiagram::datasetMarkers () const `[inherited]`

The set of dataset markers currently used, for use in legends, etc.

Note:

Cell-level override markers, if set, take precedence over the dataset values, so you might need to check these too, in order to find the marker, that is shown for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 917 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
                                                                                  {
        DataValueAttributes a =
            qVariantValue<DataValueAttributes>( attributesModel()->headerData( i, Qt::Vertical, DataVa
        const MarkerAttributes &ma = a.markerAttributes();
        ret << ma;
    }
    return ret;
}

bool AbstractDiagram::checkInvariants( bool justReturnTheStatus ) const
```

7.35.3.25 QList< QPen > AbstractDiagram::datasetPens () const `[inherited]`

The set of dataset pens currently used, for use in legends, etc.

Note:

Cell-level override pens, if set, take precedence over the dataset values, so you might need to check these too, in order to find the pens, that is used for a single cell.

Returns:

The current set of dataset pens.

Definition at line 906 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
                                                                                  {
        QPen pen = qVariantValue<QPen>( attributesModel()->headerData( i, Qt::Vertical, DatasetPenRole
        ret << pen;
    }
    return ret;
}

QList<MarkerAttributes> AbstractDiagram::datasetMarkers() const
```

7.35.3.26 DataValueAttributes AbstractDiagram::dataValueAttributes (const QModelIndex & *index*) const `[inherited]`

Retrieve the DataValueAttributes for the given index.

This will fall back automatically to what was set at dataset or model level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the attributes for.

Returns:

The DataValueAttributes for the given index.

Definition at line 427 of file KDChartAbstractDiagram.cpp.

```
{
```

7.35.3.27 DataValueAttributes AbstractDiagram::dataValueAttributes (int *column*) const `[inherited]`

Retrieve the DataValueAttributes for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the attributes for.

Returns:

The DataValueAttributes for the given dataset.

Definition at line 420 of file KDChartAbstractDiagram.cpp.

```
426 {
```

7.35.3.28 DataValueAttributes AbstractDiagram::dataValueAttributes () const `[inherited]`

Retrieve the DataValueAttributes speficied globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The global DataValueAttributes.

Definition at line 414 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::AbstractDiagram::paintDataValueText(), and KDChart::AbstractDiagram::paintMarker().

```
419 {
```

7.35.3.29 void AbstractDiagram::doItemsLayout () `[virtual, inherited]`

[reimplemented]

Definition at line 329 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::AbstractDiagram::update().

```
329                     {
330         d->plane->layoutDiagrams();
331         update();
332     }
333     QAbstractItemView::doItemsLayout();
334 }
335 
336 void AbstractDiagram::dataChanged( const QModelIndex &topLeft,
```

7.35.3.30 LineAttributes::MissingValuesPolicy LineDiagram::getCellValues (int *row*, int *column*, bool *shiftCountedXValuesByHalfSection*, double & *valueX*, double & *valueY*) const `[protected]`

Definition at line 398 of file KDChartLineDiagram.cpp.

References KDChart::AbstractDiagram::datasetDimension(), lineAttributes(), KDChart::LineAttributes::missingValuesPolicy(), and valueForCellTesting().

Referenced by paint().

```
{
    LineAttributes::MissingValuesPolicy policy;

    bool bOK = true;
    valueX = ( datasetDimension() > 1 && column > 0 )
           ? valueForCellTesting( row, column-1, bOK, true )
           : ((shiftCountedXValuesByHalfSection ? 0.5 : 0.0) + row);
    if( bOK )
        valueY = valueForCellTesting( row, column, bOK, true );
    if( bOK ){
        policy = LineAttributes::MissingValuesPolicyIgnored;
    }else{
        // missing value: find out the policy
        QModelIndex index = model()->index( row, column, rootIndex() );
        LineAttributes la = lineAttributes( index );
        policy = la.missingValuesPolicy();
    }
    return policy;
}
```

7.35.3.31 int AbstractDiagram::horizontalOffset () const `[virtual, inherited]`

[reimplemented]

Definition at line 839 of file KDChartAbstractDiagram.cpp.

```
{ return 0; }
```

7.35.3.32 QModelIndex AbstractDiagram::indexAt (const QPoint & *point*) const `[virtual, inherited]`

[reimplemented]

Definition at line 833 of file KDChartAbstractDiagram.cpp.

```
{ return QModelIndex(); }
```

7.35.3.33 bool AbstractDiagram::isHidden (const QModelIndex & *index*) const `[inherited]`

Retrieve the hidden status for the given index.

This will fall back automatically to what was set at dataset or diagram level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the hidden status for.

Returns:

The hidden status for the given index.

Definition at line 386 of file KDChartAbstractDiagram.cpp.

7.35.3.34 bool AbstractDiagram::isHidden (int *column*) const `[inherited]`

Retrieve the hidden status for the given dataset.

This will fall back automatically to what was set at diagram level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the hidden status for.

Returns:

The hidden status for the given dataset.

Definition at line 379 of file KDChartAbstractDiagram.cpp.

```
385 {
```

7.35.3.35 bool AbstractDiagram::isHidden () const `[inherited]`

Retrieve the hidden status speficied globally.

This will fall back automatically to the default settings (= not hidden), if there are no specific settings.

Returns:

The global hidden status.

Definition at line 373 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), paint(), and valueForCellTesting().

```
378 {
```

7.35.3.36 bool AbstractDiagram::isIndexHidden (const QModelIndex & *index*) const `[virtual, inherited]`

[reimplemented]

Definition at line 845 of file KDChartAbstractDiagram.cpp.

```
847 {}
```

7.35.3.37 QStringList AbstractDiagram::itemRowLabels () const `[inherited]`

The set of item row labels currently displayed, for use in Abscissa axes, etc.

Returns:
The set of item row labels currently displayed.

Definition at line 870 of file KDChartAbstractDiagram.cpp.

```
871                                                   : " << attributesModel()->rowCount(attributesModelRoo
872     const int rowCount = attributesModel()->rowCount(attributesModelRootIndex());
873     for( int i = 0; i < rowCount; ++i ){
874         //qDebug() << "item row label: " << attributesModel()->headerData( i, Qt::Vertical, Qt::Displa
875         ret << attributesModel()->headerData( i, Qt::Vertical, Qt::DisplayRole ).toString();
876     }
877     return ret;
878 }
879 
880 QStringList AbstractDiagram::datasetLabels() const
```

7.35.3.38 void KDChart::AbstractCartesianDiagram::layoutPlanes () `[virtual, inherited]`

Definition at line 113 of file KDChartAbstractCartesianDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane(), and KDChart::AbstractCoordinatePlane::layoutPlanes().

Referenced by KDChart::AbstractCartesianDiagram::addAxis(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
114 {
115     //qDebug() << "KDChart::AbstractCartesianDiagram::layoutPlanes()";
116     AbstractCoordinatePlane* plane = coordinatePlane();
117     if( plane ){
118         plane->layoutPlanes();
119         //qDebug() << "KDChart::AbstractCartesianDiagram::layoutPlanes() OK";
120     }
121 }
```

7.35.3.39 LineAttributes LineDiagram::lineAttributes (const QModelIndex & *index*) const

Definition at line 182 of file KDChartLineDiagram.cpp.

References d.

```
184 {
185     return qVariantValue<LineAttributes>(
186         d->attributesModel->data(
187             d->attributesModel->mapFromSource(index),
188             KDChart::LineAttributesRole ) );
189 }
```

7.35.3.40 LineAttributes LineDiagram::lineAttributes (int *column*) const

Definition at line 174 of file KDChartLineDiagram.cpp.

References d.

```
175 {
176     return qVariantValue<LineAttributes>(
177         d->attributesModel->data(
178             d->attributesModel->mapFromSource( columnToIndex( column ) ),
179             KDChart::LineAttributesRole ) );
180 }
```

7.35.3.41 LineAttributes LineDiagram::lineAttributes () const

Definition at line 168 of file KDChartLineDiagram.cpp.

References d.

Referenced by getCellValues(), and paint().

```
169 {
170     return qVariantValue<LineAttributes>(
171         d->attributesModel->data( KDChart::LineAttributesRole ) );
172 }
```

7.35.3.42 void KDChart::AbstractDiagram::modelsChanged () `[protected, inherited]`

This signal is emitted, when either the model or the AttributesModel is replaced.

Referenced by KDChart::AbstractDiagram::setAttributesModel(), and KDChart::AbstractDiagram::setModel().

7.35.3.43 QModelIndex AbstractDiagram::moveCursor (CursorAction *cursorAction*, Qt::KeyboardModifiers *modifiers*) `[virtual, inherited]`

[reimplemented]

Definition at line 836 of file KDChartAbstractDiagram.cpp.

```
838 { return 0; }
```

7.35.3.44 const int LineDiagram::numberOfAbscissaSegments () const `[virtual]`

Implements KDChart::AbstractCartesianDiagram.

Definition at line 899 of file KDChartLineDiagram.cpp.

References KDChart::AbstractDiagram::attributesModelRootIndex(), and d.

```
900 {
901     return d->attributesModel->rowCount(attributesModelRootIndex());
902 }
```

7.35.3.45 const int LineDiagram::numberOfOrdinateSegments () const `[virtual]`

Implements KDChart::AbstractCartesianDiagram.

Definition at line 904 of file KDChartLineDiagram.cpp.

References KDChart::AbstractDiagram::attributesModelRootIndex(), and d.

```
{
    return d->attributesModel->columnCount(attributesModelRootIndex());
}
```

7.35.3.46 void LineDiagram::paint (PaintContext ∗ *paintContext*) `[protected, virtual]`

Draw the diagram contents to the rectangle and painter, that are passed in as part of the paint context.

Parameters:

paintContext All information needed for painting.

Implements KDChart::AbstractDiagram.

Definition at line 427 of file KDChartLineDiagram.cpp.

References KDChart::AbstractDiagram::antiAliasing(), KDChart::AbstractDiagram::attributesModelRootIndex(), KDChart::AbstractDiagram::brush(), KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractDiagram::coordinatePlane(), d, KDChart::AbstractDiagram::dataBoundaries(), KDChart::AbstractDiagram::datasetDimension(), KDChart::AbstractThreeDAttributes::depth(), KDChart::LineAttributes::displayArea(), getCellValues(), KDChart::AbstractThreeDAttributes::isEnabled(), KDChart::AbstractDiagram::isHidden(), lineAttributes(), KDChart::PaintContext::painter(), KDChart::AbstractDiagram::pen(), KDChart::AbstractCartesianDiagram::referenceDiagram(), threeDLineAttributes(), KDChart::AbstractCoordinatePlane::translate(), KDChart::LineAttributes::transparency(), type(), and KDChart::AbstractDiagram::valueForCell().

Referenced by paintEvent().

```
{
//qDebug() << "    start diag::paint()";
    // note: Not having any data model assigned is no bug
    //       but we can not draw a diagram then either.
    if ( !checkInvariants( true ) ) return;
    if ( !AbstractGrid::isBoundariesValid(dataBoundaries()) ) return;

    // Make sure counted x values (== in diagrams with 1-dimensional data cells)
    // get shifted by 0.5, if the diagram's reference diagram is a BarDiagram.
    // So we get the lines to start/end at the middle of the respective bar groups.
    const bool shiftCountedXValuesByHalfSection =
            (dynamic_cast< BarDiagram* >( referenceDiagram() ) != 0);

    //QTime t = QTime::currentTime();

    const QPair<QPointF, QPointF> boundaries = dataBoundaries();
    const QPointF bottomLeft = boundaries.first;
    const QPointF topRight = boundaries.second;

    int maxFound = 0;
    {   // find the last column number that is not hidden
        const int columnCount = d->attributesModel->columnCount(attributesModelRootIndex());
        for( int iColumn =  datasetDimension()-1;
            iColumn <  columnCount;
            iColumn += datasetDimension() )
            if( ! isHidden( iColumn ) )
                maxFound = iColumn;
    }
    const int lastVisibleColumn = maxFound;
    const int rowCount = d->attributesModel->rowCount(attributesModelRootIndex());

    DataValueTextInfoList list;
    LineAttributesInfoList lineList;
    LineAttributes::MissingValuesPolicy policy;

    // paint different line types Normal - Stacked - Percent - Default Normal
    switch ( type() )
    {
        case LineDiagram::Normal:
        {
            for( int iColumn  = datasetDimension()-1;
                     iColumn <= lastVisibleColumn;
                     iColumn += datasetDimension() ) {

                //display area can be set by dataset ( == column) and/or by cell
                LineAttributes laPreviousCell; // by default no area is drawn
                QModelIndex indexPreviousCell;
                QList<QPolygonF> areas;

                bool bValuesFound = false;
                double lastValueX, lastValueY;
                double valueX, valueY;
                for ( int iRow = 0; iRow < rowCount; ++iRow ) {
                    bool skipThisCell = false;
                    // trying to find a fromPoint
                    policy = getCellValues( iRow, iColumn,
                                            shiftCountedXValuesByHalfSection,
                                            valueX, valueY );
                    switch( policy ){
                        case LineAttributes::MissingValuesAreBridged:
                            if( bValuesFound ){
                                valueX = lastValueX;
                                valueY = lastValueY;
                            }else{
                                skipThisCell = true;
                            }
                            break;
                        case LineAttributes::MissingValuesHideSegments:
                            skipThisCell = true;
                            break;
                        case LineAttributes::MissingValuesShownAsZero:
                            // fall through intended
                        case LineAttributes::MissingValuesPolicyIgnored:
                            lastValueX = valueX;
                            lastValueY = valueY;
                            bValuesFound = true;
                            break;
                    }
                    if( ! skipThisCell ){
                        // trying to find a toPoint
                        double nextValueX, nextValueY;
                        bool foundToPoint = false;
                        int iNextRow = iRow+1;
                        while ( ! (foundToPoint || skipThisCell || iNextRow >= rowCount) ) {
                            policy = getCellValues(
                                    iNextRow, iColumn,
                                    shiftCountedXValuesByHalfSection,
                                    nextValueX, nextValueY );
                            switch( policy ){
                                case LineAttributes::MissingValuesAreBridged:
                                    // The cell has no valid value, so we  make sure that
                                    // this cell will not be processed by the next iteration
                                    // of the iRow loop:
                                    ++iRow;
                                    break;
                                case LineAttributes::MissingValuesHideSegments:
                                    // The cell has no valid value, so we  make sure that
                                    // this cell will not be processed by the next iteration
                                    // of the iRow loop:
                                    skipThisCell = true;
                                    ++iRow;
                                    break;
                                case LineAttributes::MissingValuesShownAsZero:
                                    // fall through intended
                                case LineAttributes::MissingValuesPolicyIgnored:
                                    foundToPoint = true;
                                    break;
                            }
                            ++iNextRow;
                        }
                        if( ! skipThisCell ){
                            const bool isPositive = (valueY >= 0.0);
                            const QModelIndex index = model()->index( iRow, iColumn, rootIndex() );
                            const LineAttributes laCell = lineAttributes( index );
                            const bool bDisplayCellArea = laCell.displayArea();

                            QPointF fromPoint = coordinatePlane()->translate( QPointF( valueX, valueY

                            const QPointF ptNorthWest(
                                (bDisplayCellArea && ! isPositive)
                                ? coordinatePlane()->translate( QPointF( valueX, 0.0 ) )
                                : fromPoint );
                            const QPointF ptSouthWest(
                                (bDisplayCellArea && isPositive)
                                ? coordinatePlane()->translate( QPointF( valueX, 0.0 ) )
                                : fromPoint );
                            //qDebug() << "--> ptNorthWest:" << ptNorthWest;
                            //qDebug() << "--> ptSouthWest:" << ptSouthWest;
                            QPointF ptNorthEast;
                            QPointF ptSouthEast;

                            if( foundToPoint ){
                                QPointF toPoint = coordinatePlane()->translate( QPointF( nextValueX, r
                                lineList.append( LineAttributesInfo( index, fromPoint, toPoint ) );
                                ptNorthEast =
                                    (bDisplayCellArea && ! isPositive)
                                    ? coordinatePlane()->translate( QPointF( nextValueX, 0.0 ) )
                                    : toPoint;
                                ptSouthEast =
                                    (bDisplayCellArea && isPositive)
                                    ? coordinatePlane()->translate( QPointF( nextValueX, 0.0 ) )
                                    : toPoint;
                                // we can’t take as a condition the line attributes
                                // to be different from a cell to another.
                                // e.g the user should be able to have different brush
                                // which is not reflected in the line attributes
                                // see examples/Area which show such an option
                                if( areas.count() /*&& laCell != laPreviousCell*/ ){
                                    paintAreas( ctx, indexPreviousCell, areas, laPreviousCell.transpar
                                    areas.clear();
                                }
                                if( bDisplayCellArea ){
                                    QPolygonF poly;
                                    poly << ptNorthWest << ptNorthEast << ptSouthEast << ptSouthWest;
                                    //qDebug() << "ptNorthWest:" << ptNorthWest;
                                    //qDebug() << "ptNorthEast:" << ptNorthEast;
                                    //qDebug() << "ptSouthEast:" << ptSouthEast;
                                    //qDebug() << "ptSouthWest:" << ptSouthWest;
                                    //qDebug() << "polygon:" << poly;
```

```
                                    areas << poly;
                                    laPreviousCell = laCell;
                                    indexPreviousCell = index;
                                }
                            }else{
                                ptNorthEast = ptNorthWest;
                                ptSouthEast = ptSouthWest;
                            }

                            const PositionPoints pts( ptNorthWest, ptNorthEast, ptSouthEast, ptSouthWe
                            d->appendDataValueTextInfoToList( this, list, index, pts,
                                    Position::NorthWest, Position::SouthWest,
                                    valueY );
                        }
                    }
                }
                if( areas.count() ){
                    paintAreas( ctx, indexPreviousCell, areas, laPreviousCell.transparency() );
                    areas.clear();
                }
            }
        }
            break;
        case LineDiagram::Stacked:
            // fall-through intended
        case LineDiagram::Percent:
        {
            //FIXME(khz): add LineAttributes::MissingValuesPolicy support for LineDiagram::Stacked and

            const bool isPercentMode = type() == LineDiagram::Percent;
            double maxValue = 100; // always 100%
            double sumValues = 0;
            QVector <double > percentSumValues;

            //calculate sum of values for each column and store
            if( isPercentMode ){
                for ( int j=0; j<rowCount ; ++j ) {
                    for( int i =  datasetDimension()-1;
                        i <= lastVisibleColumn;
                        i += datasetDimension() ) {
                            double tmpValue = valueForCell( j, i );
                            if ( tmpValue > 0 )
                                sumValues += tmpValue;
                            if ( i == lastVisibleColumn ) {
                                percentSumValues <<  sumValues ;
                                sumValues = 0;
                            }
                        }
                }
            }

            QList<QPointF> bottomPoints;
            bool bFirstDataset = true;

            for( int iColumn =  datasetDimension()-1;
                 iColumn <= lastVisibleColumn;
                 iColumn += datasetDimension() ) {

                //display area can be set by dataset ( == column) and/or by cell
                LineAttributes laPreviousCell; // by default no area is drawn
                QModelIndex indexPreviousCell;
                QList<QPolygonF> areas;
                QList<QPointF> points;

                for ( int iRow = 0; iRow< rowCount; ++iRow ) {
                    const QModelIndex index = model()->index( iRow, iColumn, rootIndex() );
                    const LineAttributes laCell = lineAttributes( index );
                    const bool bDisplayCellArea = laCell.displayArea();

                    double stackedValues = 0, nextValues = 0;
                    for ( int iColumn2 = iColumn;
                          iColumn2 >= datasetDimension()-1;
                          iColumn2 -= datasetDimension() )
                    {
                        const double val = valueForCell( iRow, iColumn2 );
                        if( val > 0 || ! isPercentMode )
                            stackedValues += val;
                        //qDebug() << valueForCell( iRow, iColumn2 );
                        if ( iRow+1 < rowCount ){
                            const double val = valueForCell( iRow+1, iColumn2 );
                            if( val > 0 || ! isPercentMode )
                                nextValues += val;
                        }
                    }
                    if( isPercentMode ){
                        if ( percentSumValues.at( iRow ) != 0  )
                            stackedValues = stackedValues / percentSumValues.at( iRow ) * maxValue;
                        else
                            stackedValues = 0.0;
                    }
                    //qDebug() << stackedValues << endl;
                    QPointF nextPoint = coordinatePlane()->translate( QPointF( iRow, stackedValues ) )
                    points << nextPoint;

                    const QPointF ptNorthWest( nextPoint );
                    const QPointF ptSouthWest(
                            bDisplayCellArea
                            ? ( bFirstDataset
                                ? coordinatePlane()->translate( QPointF( iRow, 0.0 ) )
                                : bottomPoints.at( iRow )
                              )
                            : nextPoint );
                    QPointF ptNorthEast;
                    QPointF ptSouthEast;

                    if ( iRow+1 < rowCount ){
                        if( isPercentMode ){
                            if ( percentSumValues.at( iRow+1 ) != 0  )
                                nextValues = nextValues / percentSumValues.at( iRow+1 ) * maxValue;
                            else
                                nextValues = 0.0;
                        }
                        QPointF toPoint = coordinatePlane()->translate( QPointF( iRow+1, nextValues )
                        lineList.append( LineAttributesInfo( index, nextPoint, toPoint ) );
                        ptNorthEast = toPoint;
                        ptSouthEast =
                            bDisplayCellArea
                            ? ( bFirstDataset
                                ? coordinatePlane()->translate( QPointF( iRow+1, 0.0 ) )
                                : bottomPoints.at( iRow+1 )
                              )
                            : toPoint;
                        if( areas.count() && laCell != laPreviousCell ){
                            paintAreas( ctx, indexPreviousCell, areas, laPreviousCell.transparency() )
                            areas.clear();
                        }
                        if( bDisplayCellArea ){
                            QPolygonF poly;
                            poly << ptNorthWest << ptNorthEast << ptSouthEast << ptSouthWest;
                            areas << poly;
                            laPreviousCell = laCell;
                            indexPreviousCell = index;
                        }else{
                            //qDebug() << "no area shown for row"<<iRow<<"  column"<<iColumn;
                        }
                    }else{
                        ptNorthEast = ptNorthWest;
                        ptSouthEast = ptSouthWest;
                    }

                    const PositionPoints pts( ptNorthWest, ptNorthEast, ptSouthEast, ptSouthWest );
                    d->appendDataValueTextInfoToList( this, list, index, pts,
                            Position::NorthWest, Position::SouthWest,
                            valueForCell( iRow, iColumn ) );
                }
                if( areas.count() ){
                    paintAreas( ctx, indexPreviousCell, areas, laPreviousCell.transparency() );
                    areas.clear();
                }
                bottomPoints = points;
                bFirstDataset = false;
            }
        }
            break;
        default:
            Q_ASSERT_X ( false, "paint()",
                         "Type item does not match a defined line chart Type." );
    }
    // paint all lines and their attributes
    {
        PainterSaver painterSaver( ctx->painter() );
        if ( antiAliasing() )
            ctx->painter()->setRenderHint ( QPainter::Antialiasing );
        LineAttributesInfoListIterator itline ( lineList );

        //qDebug() << "Rendering 1 in: " << t.msecsTo( QTime::currentTime() ) << endl;

        QBrush curBrush;
        QPen curPen;
        QPolygonF points;
        while ( itline.hasNext() ) {
            const LineAttributesInfo& lineInfo = itline.next();
            const QModelIndex& index = lineInfo.index;
            const ThreeDLineAttributes td = threeDLineAttributes( index );
            if( td.isEnabled() ){
                paintThreeDLines( ctx, index, lineInfo.value, lineInfo.nextValue, td.depth() );
            }else{
                const QBrush br( brush( index ) );
                const QPen   pn( pen(   index ) );
                if( points.count() && points.last() == lineInfo.value && curBrush == br && curPen == p
                    points << lineInfo.nextValue;
                }else{
                    if( points.count() )
                        paintPolyline( ctx, curBrush, curPen, points );
                    curBrush = br;
                    curPen   = pn;
                    points.clear();
                    points << lineInfo.value << lineInfo.nextValue;
                }
            }
        }
        if( points.count() )
            paintPolyline( ctx, curBrush, curPen, points );
    }
    // paint all data value texts and the point markers
    d->paintDataValueTextsAndMarkers( this, ctx, list, true );
    //qDebug() << "Rendering 2 in: " << t.msecsTo( QTime::currentTime() ) << endl;
}
```

7.35.3.47 void AbstractDiagram::paintDataValueText (QPainter ∗ *painter*, const QModelIndex & *index*, const QPointF & *pos*, double *value*) `[inherited]`

Definition at line 474 of file KDChartAbstractDiagram.cpp.

References KDChart::RelativePosition::alignment(), KDChart::TextAttributes::calculatedFont(), d, KDChart::DataValueAttributes::dataLabel(), KDChart::AbstractDiagram::dataValueAttributes(), KDChart::DataValueAttributes::decimalDigits(), KDChart::TextAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::TextAttributes::pen(), KDChart::DataValueAttributes::position(), KDChart::DataValueAttributes::prefix(), KDChart::TextAttributes::rotation(), KDChart::DataValueAttributes::showRepetitiveDataLabels(), KDChart::DataValueAttributes::suffix(), and KDChart::DataValueAttributes::textAttributes().

Referenced by KDChart::RingDiagram::paint(), and KDChart::PolarDiagram::paint().

```
{
    // paint one data series
    const DataValueAttributes a( dataValueAttributes(index) );
    if ( !a.isVisible() ) return;

    // handle decimal digits
    int decimalDigits = a.decimalDigits();
    int decimalPos = QString::number(  value ).indexOf( QLatin1Char( ’.’ ) );
    QString roundedValue;
    if ( a.dataLabel().isNull() ) {
        if ( decimalPos > 0 && value != 0 )
            roundedValue =  roundValues ( value, decimalPos, decimalDigits );
        else
            roundedValue = QString::number(  value );
    } else
        roundedValue = a.dataLabel();
        // handle prefix and suffix
    if ( !a.prefix().isNull() )
        roundedValue.prepend( a.prefix() );

    if ( !a.suffix().isNull() )
        roundedValue.append( a.suffix() );

    const TextAttributes ta( a.textAttributes() );
    // FIXME draw the non-text bits, background, etc
    if ( ta.isVisible() ) {

        QPointF pt( pos );
        /* for debugging:
        PainterSaver painterSaver( painter );
        painter->setPen( Qt::black );
        painter->drawLine( pos - QPointF( 1,1), pos + QPointF( 1,1) );
        painter->drawLine( pos - QPointF(-1,1), pos + QPointF(-1,1) );
        */

        // adjust the text start point position, if alignment is not Bottom/Left
        const RelativePosition relPos( a.position( value >= 0.0 ) );
        const Qt::Alignment alignBottomLeft = Qt::AlignBottom | Qt::AlignLeft;
        const QFont calculatedFont( ta.calculatedFont( d->plane, KDChartEnums::MeasureOrientationMinim
        //qDebug() << "calculatedFont’s point size:" << calculatedFont.pointSizeF();
        if( (relPos.alignment() & alignBottomLeft) != alignBottomLeft ){
            const QRectF boundRect(
                    d->cachedFontMetrics( calculatedFont, this )->boundingRect( roundedValue ) );
            if( relPos.alignment() & Qt::AlignRight )
                pt.rx() -= boundRect.width();
            else if( relPos.alignment() & Qt::AlignHCenter )
                pt.rx() -= 0.5 * boundRect.width();

            if( relPos.alignment() & Qt::AlignTop )
                pt.ry() += boundRect.height();
            else if( relPos.alignment() & Qt::AlignVCenter )
                pt.ry() += 0.5 * boundRect.height();
        }

        // FIXME draw the non-text bits, background, etc

        if ( a.showRepetitiveDataLabels() ||
             pos.x() <= d->lastX ||
             d->lastRoundedValue != roundedValue ) {
            d->lastRoundedValue = roundedValue;
            d->lastX = pos.x();

            PainterSaver painterSaver( painter );
            painter->setPen( ta.pen() );
            painter->setFont( calculatedFont );
            painter->translate( pt );
            painter->rotate( ta.rotation() );
            painter->drawText( QPointF(0, 0), roundedValue );
        }
    }
}
```

7.35.3.48 void AbstractDiagram::paintDataValueTexts (QPainter * *painter*) `[protected, virtual, inherited]`

Definition at line 576 of file KDChartAbstractDiagram.cpp.

```
                                                                                {
       for ( int j=0; j< rowCount; ++j ) {
           const QModelIndex index = model()->index( j, i, rootIndex() );
           double value = model()->data( index ).toDouble();
           const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
           paintDataValueText( painter, index, pos, value );
       }
    }
}
```

7.35.3.49 void LineDiagram::paintEvent (QPaintEvent *) `[protected]`

Definition at line 371 of file KDChartLineDiagram.cpp.

References paint(), KDChart::PaintContext::setPainter(), and KDChart::PaintContext::setRectangle().

```
{
//qDebug() << "starting LineDiagram::paintEvent ( QPaintEvent*)";
    QPainter painter ( viewport() );
    PaintContext ctx;
    ctx.setPainter ( &painter );
    ctx.setRectangle ( QRectF ( 0, 0, width(), height() ) );
    paint ( &ctx );
//qDebug() << "         LineDiagram::paintEvent ( QPaintEvent*) ended.";
}
```

7.35.3.50 void AbstractDiagram::paintMarker (QPainter ∗ *painter*, const QModelIndex & *index*, const QPointF & *pos*) `[inherited]`

Definition at line 592 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::brush(), KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractDiagram::dataValueAttributes(), KDChart::MarkerAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::DataValueAttributes::markerAttributes(), KDChart::MarkerAttributes::markerColor(), KDChart::MarkerAttributes::markerSize(), KDChart::AbstractDiagram::paintMarker(), and KDChart::MarkerAttributes::pen().

```
{

    if ( !checkInvariants() ) return;
    DataValueAttributes a = dataValueAttributes(index);
    if ( !a.isVisible() ) return;
    const MarkerAttributes &ma = a.markerAttributes();
    if ( !ma.isVisible() ) return;

    PainterSaver painterSaver( painter );
    QSizeF maSize( ma.markerSize() );
    QBrush indexBrush( brush( index ) );
    QPen indexPen( ma.pen() );
    if ( ma.markerColor().isValid() )
        indexBrush.setColor( ma.markerColor() );

    paintMarker( painter, ma, indexBrush, indexPen, pos, maSize );
}


```

7.35.3.51 void AbstractDiagram::paintMarker (QPainter ∗ *painter*, const MarkerAttributes & *markerAttributes*, const QBrush & *brush*, const QPen &, const QPointF & *point*, const QSizeF & *size*) `[virtual, inherited]`

Definition at line 614 of file KDChartAbstractDiagram.cpp.

References KDChart::MarkerAttributes::markerStyle().

Referenced by KDChart::MarkerLayoutItem::paintIntoRect(), and KDChart::AbstractDiagram::paintMarker().

```
{

    const QPen oldPen( painter->pen() );
    // Pen is used to paint 4Pixels - 1 Pixel - Ring and FastCross types.
    // make sure to use the brush color - see above in those cases.
    const bool isFourPixels = (markerAttributes.markerStyle() == MarkerAttributes::Marker4Pixels);
    if( isFourPixels || (markerAttributes.markerStyle() == MarkerAttributes::Marker1Pixel) ){
        // for high-performance point charts with tiny point markers:
        painter->setPen( QPen( brush.color().light() ) );
        if( isFourPixels ){
            const qreal x = pos.x();
            const qreal y = pos.y();
            painter->drawLine( QPointF(x-1.0,y-1.0),
                               QPointF(x+1.0,y-1.0) );
            painter->drawLine( QPointF(x-1.0,y),
                               QPointF(x+1.0,y) );
            painter->drawLine( QPointF(x-1.0,y+1.0),
                               QPointF(x+1.0,y+1.0) );
        }
        painter->drawPoint( pos );
    }else{
        PainterSaver painterSaver( painter );
        // we only a solid line surrounding the markers
        QPen painterPen( pen );
        painterPen.setStyle( Qt::SolidLine );
        painter->setPen( painterPen );
        painter->setBrush( brush );
        painter->setRenderHint ( QPainter::Antialiasing );
        painter->translate( pos );
        switch ( markerAttributes.markerStyle() ) {
            case MarkerAttributes::MarkerCircle:
                painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                            maSize.height(), maSize.width()) );
                break;
            case MarkerAttributes::MarkerSquare:
                {
                    QRectF rect( 0 - maSize.width()/2, 0 - maSize.height()/2,
                                maSize.width(), maSize.height() );
                    painter->drawRect( rect );
                    painter->fillRect( rect, brush.color() );
                    break;
                }
            case MarkerAttributes::MarkerDiamond:
                {
                    QVector <QPointF > diamondPoints;
                    QPointF top, left, bottom, right;
                    top    = QPointF( 0, 0 - maSize.height()/2 );
                    left   = QPointF( 0 - maSize.width()/2, 0 );
                    bottom = QPointF( 0, maSize.height()/2 );
                    right  = QPointF( maSize.width()/2, 0 );
                    diamondPoints << top << left << bottom << right;
                    painter->drawPolygon( diamondPoints );
                    break;
                }
            // both handled on top of the method:
            case MarkerAttributes::Marker1Pixel:
            case MarkerAttributes::Marker4Pixels:
                    break;
            case MarkerAttributes::MarkerRing:
                {
                    painter->setPen( QPen( brush.color() ) );
                    painter->setBrush( Qt::NoBrush );
                    painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                                        maSize.height(), maSize.width()) );
                    break;
                }
            case MarkerAttributes::MarkerCross:
                {
                    QRectF rect( maSize.width()*-0.5, maSize.height()*-0.2,
                                 maSize.width(), maSize.height()*0.4 );
                    painter->drawRect( rect );
                    rect.setTopLeft(QPointF( maSize.width()*-0.2, maSize.height()*-0.5 ));
                    rect.setSize(QSizeF( maSize.width()*0.4, maSize.height() ));
                    painter->drawRect( rect );
                    break;
                }
            case MarkerAttributes::MarkerFastCross:
                {
                    QPointF left, right, top, bottom;
                    left  = QPointF( -maSize.width()/2, 0 );
                    right = QPointF( maSize.width()/2, 0 );
                    top   = QPointF( 0, -maSize.height()/2 );
                    bottom= QPointF( 0, maSize.height()/2 );
                    painter->setPen( QPen( brush.color() ) );
                    painter->drawLine( left, right );
                    painter->drawLine(  top, bottom );
                    break;
                }
            default:
                Q_ASSERT_X ( false, "paintMarkers()",
                            "Type item does not match a defined Marker Type." );
        }
    }
    painter->setPen( oldPen );
}

void AbstractDiagram::paintMarkers( QPainter* painter )
```

7.35.3.52 void AbstractDiagram::paintMarkers (QPainter ∗ *painter*) [protected, virtual, inherited]

Definition at line 716 of file KDChartAbstractDiagram.cpp.

```
                                                                               {
       for ( int j=0; j< rowCount; ++j ) {
           const QModelIndex index = model()->index( j, i, rootIndex() );
           double value = model()->data( index ).toDouble();
           const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
           paintMarker( painter, index, pos );
       }
    }
}


```

7.35.3.53 QPen AbstractDiagram::pen (const QModelIndex & *index*) const [inherited]

Retrieve the pen to be used, for painting the datapoint at the given index in the model.

Parameters:

index The index of the datapoint in the model.

Returns:

The pen to use for painting.

Definition at line 770 of file KDChartAbstractDiagram.cpp.

```
{
```

7.35.3.54 QPen AbstractDiagram::pen (int *dataset*) const [inherited]

Retrieve the pen to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the pen for.

Returns:

The pen to use for painting.

Definition at line 762 of file KDChartAbstractDiagram.cpp.

```
769 {
```

7.35.3.55 QPen AbstractDiagram::pen () const `[inherited]`

Retrieve the pen to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:
The pen to use for painting.

Definition at line 756 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::PieDiagram::paint(), and paint().

```
761 {
```

7.35.3.56 bool AbstractDiagram::percentMode () const `[inherited]`

Definition at line 468 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::CartesianCoordinatePlane::getDataDimensionsList().

7.35.3.57 void KDChart::AbstractDiagram::propertiesChanged () `[protected, inherited]`

Emitted upon change of a property of the Diagram.

Referenced by resetLineAttributes(), KDChart::AbstractDiagram::setDataValueAttributes(), setLineAttributes(), setThreeDLineAttributes(), and setType().

7.35.3.58 AbstractCartesianDiagram ∗ AbstractCartesianDiagram::referenceDiagram () const `[virtual, inherited]`

Definition at line 146 of file KDChartAbstractCartesianDiagram.cpp.

References d.

Referenced by paint(), and referenceDiagramIsBarDiagram().

```
147 {
148     return d->referenceDiagram;
149 }
```

7.35.3.59 QPointF AbstractCartesianDiagram::referenceDiagramOffset () const `[virtual, inherited]`

Definition at line 151 of file KDChartAbstractCartesianDiagram.cpp.

References d.

```
152 {
153     return d->referenceDiagramOffset;
154 }
```

7.35.3.60 void LineDiagram::resetLineAttributes (const QModelIndex & *index*)

Remove any explicit line attributes settings that might have been specified before.

Definition at line 161 of file KDChartLineDiagram.cpp.

References d, KDChart::LineAttributesRole, and KDChart::AbstractDiagram::propertiesChanged().

```
162 {
163     d->attributesModel->resetData(
164             d->attributesModel->mapFromSource(index), LineAttributesRole );
165     emit propertiesChanged();
166 }
```

7.35.3.61 void LineDiagram::resetLineAttributes (int *column*)

Definition at line 140 of file KDChartLineDiagram.cpp.

References d, KDChart::LineAttributesRole, and KDChart::AbstractDiagram::propertiesChanged().

```
141 {
142     d->attributesModel->resetHeaderData(
143             column, Qt::Vertical, LineAttributesRole );
144     emit propertiesChanged();
145 }
```

7.35.3.62 void LineDiagram::resize (const QSizeF & *area*) `[virtual]`

Called by the widget's sizeEvent.

Adjust all internal structures, that are calculated, dependending on the size of the widget.

Parameters:

area

Implements KDChart::AbstractDiagram.

Definition at line 895 of file KDChartLineDiagram.cpp.

```
896 {
897 }
```

7.35.3.63 void LineDiagram::resizeEvent (QResizeEvent ∗) `[protected]`

Definition at line 263 of file KDChartLineDiagram.cpp.

```
264 {
265 }
```

7.35.3.64 void AbstractDiagram::scrollTo (const QModelIndex & *index*, ScrollHint *hint* = EnsureVisible) `[virtual, inherited]`

[reimplemented]

Definition at line 830 of file KDChartAbstractDiagram.cpp.

```
832 { return QModelIndex(); }
```

7.35.3.65 void AbstractDiagram::setAllowOverlappingDataValueTexts (bool *allow*) `[inherited]`

Set whether data value labels are allowed to overlap.

Parameters:

allow True means that overlapping labels are allowed.

Definition at line 440 of file KDChartAbstractDiagram.cpp.

References d.

```
445 {
```

7.35.3.66 void AbstractDiagram::setAntiAliasing (bool *enabled*) `[inherited]`

Set whether anti-aliasing is to be used while rendering this diagram.

Parameters:

enabled True means that AA is enabled.

Definition at line 451 of file KDChartAbstractDiagram.cpp.

References d.

```
456 {
```

7.35.3.67 void AbstractDiagram::setAttributesModel (AttributesModel ∗ *model*) `[virtual, inherited]`

Associate an AttributesModel with this diagram.

Note that the diagram does _not_ take ownership of the AttributesModel. This should thus only be used with AttributesModels that have been explicitely created by the user, and are owned by her. Setting an AttributesModel that is internal to another diagram is an error.

Correct:

```
AttributesModel *am = new AttributesModel( model, 0 );
diagram1->setAttributesModel( am );
diagram2->setAttributesModel( am );
```

Wrong:

```
diagram1->setAttributesModel( diagram2->attributesModel() );
```

Parameters:

model The AttributesModel to use for this diagram.

See also:

AttributesModel, usesExternalAttributesModel

Definition at line 261 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::AbstractDiagram::modelsChanged().

```
{
    if( amodel->sourceModel() != model() ) {
        qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
                 "Trying to set an attributesmodel which works on a different "
                 "model than the diagram.");
        return;
    }
    if( qobject_cast<PrivateAttributesModel*>(amodel) ) {
        qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
                 "Trying to set an attributesmodel that is private to another diagram.");
        return;
    }
    d->setAttributesModel(amodel);
    scheduleDelayedItemsLayout();
    d->databoundariesDirty = true;
    emit modelsChanged();
}
```

7.35.3.68 void AbstractDiagram::setAttributesModelRootIndex (const QModelIndex & *idx*) `[protected, inherited]`

Definition at line 301 of file KDChartAbstractDiagram.cpp.

References d.

7.35.3.69 void AbstractDiagram::setBrush (const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting all datasets in the model.

Parameters:

brush The brush to use.

Definition at line 786 of file KDChartAbstractDiagram.cpp.

```
{
```

7.35.3.70 void AbstractDiagram::setBrush (int *dataset*, const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting the given dataset.

Parameters:

dataset The dataset's column in the model.

pen The brush to use.

Definition at line 793 of file KDChartAbstractDiagram.cpp.

```
801 {
```

7.35.3.71 void AbstractDiagram::setBrush (const QModelIndex & *index*, const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting the datapoint at the given index.

Parameters:

index The datapoint’s index in the model.

brush The brush to use.

Definition at line 778 of file KDChartAbstractDiagram.cpp.

```
785 {
```

7.35.3.72 void KDChart::AbstractCartesianDiagram::setCoordinatePlane (AbstractCoordinatePlane ∗ *plane*) `[virtual, inherited]`

Set the coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. The chart takes ownership.

Returns:

The coordinate plane associated with the diagram.

Reimplemented from KDChart::AbstractDiagram.

Definition at line 123 of file KDChartAbstractCartesianDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane(), and KDChart::AbstractDiagram::setCoordinatePlane().

```
124 {
125     if( coordinatePlane() ) disconnect( coordinatePlane() );
126     AbstractDiagram::setCoordinatePlane(plane);
127
128     // show the axes, after all have been adjusted
129     // (because they might be dependend on each other)
130     /*
131     if( plane )
132         Q_FOREACH( CartesianAxis* axis, d->axesList )
133             axis->show();
134     else
135         Q_FOREACH( CartesianAxis* axis, d->axesList )
136             axis->hide();
137     */
138 }
```

7.35.3.73 void AbstractDiagram::setDataBoundariesDirty () const `[protected, inherited]`

Definition at line 240 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::BarDiagram::setThreeDBarAttributes(), setThreeDLineAttributes(), setType(), and KDChart::BarDiagram::setType().

```
241 {
242     d->databoundariesDirty = true;
243 }
```

7.35.3.74 void AbstractDiagram::setDatasetDimension (int *dimension*) `[inherited]`

Sets the dataset dimension of the diagram.

See also:

datasetDimension.

Parameters:

dimension

Definition at line 947 of file KDChartAbstractDiagram.cpp.

References d.

```
954 {
```

7.35.3.75 void AbstractDiagram::setDataValueAttributes (const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for all datapoints in the model.

Parameters:

a The attributes to set.

Definition at line 434 of file KDChartAbstractDiagram.cpp.

References d.

```
439 {
```

7.35.3.76 void AbstractDiagram::setDataValueAttributes (int *dataset*, const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for the given dataset.

Parameters:

dataset The dataset to set the attributes for.

a The attributes to set.

Definition at line 406 of file KDChartAbstractDiagram.cpp.

References d.

```
413 {
```

7.35.3.77 void AbstractDiagram::setDataValueAttributes (const QModelIndex & *index*, const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for the given index.

Parameters:

index The datapoint to set the attributes for.

a The attributes to set.

Definition at line 395 of file KDChartAbstractDiagram.cpp.

References d, KDChart::DataValueLabelAttributesRole, and KDChart::AbstractDiagram::propertiesChanged().

```
395 {
396     d->attributesModel->setData(
397         d->attributesModel->mapFromSource( index ),
398         qVariantFromValue( a ),
399         DataValueLabelAttributesRole );
400     emit propertiesChanged();
401 }
402
403
```

7.35.3.78 void AbstractDiagram::setHidden (bool *hidden*) `[inherited]`

Hide (or unhide, resp.) all datapoints in the model.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

hidden The hidden status to set.

Definition at line 365 of file KDChartAbstractDiagram.cpp.

References d.

```
372 {
```

7.35.3.79 void AbstractDiagram::setHidden (int *column*, bool *hidden*) `[inherited]`

Hide (or unhide, resp.) a dataset.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

dataset The dataset to set the hidden status for.

hidden The hidden status to set.

Definition at line 356 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.35.3.80 void AbstractDiagram::setHidden (const QModelIndex & *index*, bool *hidden*) `[inherited]`

Hide (or unhide, resp.) a data cell.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

index The datapoint to set the hidden status for.

hidden The hidden status to set.

Definition at line 347 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::DataHiddenRole.

```
{
```

7.35.3.81 void LineDiagram::setLineAttributes (const QModelIndex & *index*, const LineAttributes & *a*)

Definition at line 147 of file KDChartLineDiagram.cpp.

References d, KDChart::LineAttributesRole, and KDChart::AbstractDiagram::propertiesChanged().

```
{
    d->attributesModel->setData(
            d->attributesModel->mapFromSource(index),
    qVariantFromValue( ta ),
    LineAttributesRole );
    emit propertiesChanged();
}
```

7.35.3.82 void LineDiagram::setLineAttributes (int *column*, const LineAttributes & *a*)

Definition at line 128 of file KDChartLineDiagram.cpp.

References d, KDChart::LineAttributesRole, and KDChart::AbstractDiagram::propertiesChanged().

```
{
    d->attributesModel->setHeaderData(
            column,
            Qt::Vertical,
            qVariantFromValue( ta ),
            LineAttributesRole );
    emit propertiesChanged();
}
```

7.35.3.83 void LineDiagram::setLineAttributes (const LineAttributes & *a*)

Definition at line 120 of file KDChartLineDiagram.cpp.

References d, KDChart::LineAttributesRole, and KDChart::AbstractDiagram::propertiesChanged().

```
{
     d->attributesModel->setModelData(
        qVariantFromValue( ta ),
        LineAttributesRole );
   emit propertiesChanged();
}
```

7.35.3.84 void AbstractDiagram::setModel (QAbstractItemModel ∗ *model*) `[virtual, inherited]`

Associate a model with the diagram.

Definition at line 245 of file KDChartAbstractDiagram.cpp.

References d, KDChart::AttributesModel::initFrom(), and KDChart::AbstractDiagram::modelsChanged().

```
{
  QAbstractItemView::setModel( newModel );
  AttributesModel* amodel = new PrivateAttributesModel( newModel, this );
  amodel->initFrom( d->attributesModel );
  d->setAttributesModel(amodel);
  scheduleDelayedItemsLayout();
  d->databoundariesDirty = true;
  emit modelsChanged();
}
```

7.35.3.85 void AbstractDiagram::setPen (const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting all datasets in the model.

Parameters:

pen The pen to use.

Definition at line 740 of file KDChartAbstractDiagram.cpp.

```
{
```

7.35.3.86 void AbstractDiagram::setPen (int *dataset*, const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting the given dataset.

Parameters:

dataset The dataset's row in the model.

pen The pen to use.

Definition at line 747 of file KDChartAbstractDiagram.cpp.

```
755 {
```

7.35.3.87 void AbstractDiagram::setPen (const QModelIndex & *index*, const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting the datapoint at the given index.

Parameters:

index The datapoint's index in the model.

pen The pen to use.

Definition at line 732 of file KDChartAbstractDiagram.cpp.

```
739 {
```

7.35.3.88 void AbstractDiagram::setPercentMode (bool *percent*) `[inherited]`

Definition at line 462 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by setType(), and KDChart::BarDiagram::setType().

```
467 {
```

7.35.3.89 void AbstractCartesianDiagram::setReferenceDiagram (AbstractCartesianDiagram ∗ *diagram*, const QPointF & *offset* = QPointF()) `[virtual, inherited]`

Definition at line 140 of file KDChartAbstractCartesianDiagram.cpp.

References d.

```
141 {
142     d->referenceDiagram = diagram;
143     d->referenceDiagramOffset = offset;
144 }
```

7.35.3.90 void AbstractDiagram::setRootIndex (const QModelIndex & *idx*) `[virtual, inherited]`

Set the root index in the model, where the diagram starts referencing data for display.

[reimplemented]

Definition at line 294 of file KDChartAbstractDiagram.cpp.

References d.

7.35.3.91 void AbstractDiagram::setSelection (const QRect & *rect*, QItemSelectionModel::SelectionFlags *command*) `[virtual, inherited]`

[reimplemented]

Definition at line 848 of file KDChartAbstractDiagram.cpp.

```
{ return QRegion(); }
```

7.35.3.92 void LineDiagram::setThreeDLineAttributes (const QModelIndex & *index*, const ThreeDLineAttributes & *a*)

Definition at line 214 of file KDChartLineDiagram.cpp.

References d, KDChart::AbstractDiagram::propertiesChanged(), KDChart::AbstractDiagram::setDataBoundariesDirty(), and KDChart::ThreeDLineAttributesRole.

```
{
    setDataBoundariesDirty();
    d->attributesModel->setData(
        d->attributesModel->mapFromSource(index),
        qVariantFromValue( ta ),
        ThreeDLineAttributesRole );
   emit propertiesChanged();
}
```

7.35.3.93 void LineDiagram::setThreeDLineAttributes (int *column*, const ThreeDLineAttributes & *a*)

Definition at line 201 of file KDChartLineDiagram.cpp.

References d, KDChart::AbstractDiagram::propertiesChanged(), KDChart::AbstractDiagram::setDataBoundariesDirty(), and KDChart::ThreeDLineAttributesRole.

```
{
    setDataBoundariesDirty();
    d->attributesModel->setHeaderData(
        column,
        Qt::Vertical,
        qVariantFromValue( ta ),
        ThreeDLineAttributesRole );
   emit propertiesChanged();
}
```

7.35.3.94 void LineDiagram::setThreeDLineAttributes (const ThreeDLineAttributes & *a*)

Definition at line 191 of file KDChartLineDiagram.cpp.

References d, KDChart::AbstractDiagram::propertiesChanged(), KDChart::AbstractDiagram::setDataBoundariesDirty(), and KDChart::ThreeDLineAttributesRole.

```
{
    setDataBoundariesDirty();
    d->attributesModel->setModelData(
        qVariantFromValue( ta ),
        ThreeDLineAttributesRole );
   emit propertiesChanged();
}
```

7.35.3.95 void LineDiagram::setType (const LineType *type*)

Definition at line 99 of file KDChartLineDiagram.cpp.

References d, KDChart::AbstractDiagram::datasetDimension(), KDChart::AbstractDiagram::propertiesChanged(), KDChart::AbstractDiagram::setDataBoundariesDirty(), and KDChart::AbstractDiagram::setPercentMode().

```
{
   if ( d->lineType == type ) return;
   if ( type != LineDiagram::Normal && datasetDimension() > 1 ) {
       Q_ASSERT_X ( false, "setType()",
                    "This line chart type can’t be used with multi-dimensional data." );
       return;
   }
   d->lineType = type;
   // AbstractAxis settings - see AbstractDiagram and CartesianAxis
   setPercentMode( type == LineDiagram::Percent );
   setDataBoundariesDirty();
   emit layoutChanged( this );
   emit propertiesChanged();
}
```

7.35.3.96 void AbstractCartesianDiagram::takeAxis (CartesianAxis ∗ *axis*) `[virtual, inherited]`

Removes the axis from the diagram, without deleting it.

The diagram no longer owns the axis, so it is the caller’s responsibility to delete the axis.

See also:
addAxis

Definition at line 98 of file KDChartAbstractCartesianDiagram.cpp.

References d, KDChart::AbstractAxis::deleteObserver(), KDChart::AbstractCartesianDiagram::layoutPlanes(), and KDChart::AbstractLayoutItem::setParentWidget().

Referenced by KDChart::CartesianAxis::∼CartesianAxis().

```
{
     const int idx = d->axesList.indexOf( axis );
101     if( idx != -1 )
102         d->axesList.takeAt( idx );
103     axis->deleteObserver( this );
104     axis->setParentWidget( 0 );
105     layoutPlanes();
106 }
```

7.35.3.97 double LineDiagram::threeDItemDepth (int *column*) const `[protected, virtual]`

Implements KDChart::AbstractCartesianDiagram.

Definition at line 254 of file KDChartLineDiagram.cpp.

References d.

```
255 {
256     return qVariantValue<ThreeDLineAttributes>(
257         d->attributesModel->headerData (
258             column,
259             Qt::Vertical,
260             KDChart::ThreeDLineAttributesRole ) ).validDepth();
261 }
```

7.35.3.98 double LineDiagram::threeDItemDepth (const QModelIndex & *index*) const `[protected, virtual]`

Implements KDChart::AbstractCartesianDiagram.

Definition at line 249 of file KDChartLineDiagram.cpp.

References threeDLineAttributes(), and KDChart::AbstractThreeDAttributes::validDepth().

```
250 {
251     return threeDLineAttributes( index ).validDepth();
252 }
```

7.35.3.99 ThreeDLineAttributes LineDiagram::threeDLineAttributes (const QModelIndex & *index*) const

Definition at line 240 of file KDChartLineDiagram.cpp.

References d.

```
242 {
243     return qVariantValue<ThreeDLineAttributes>(
244         d->attributesModel->data(
245             d->attributesModel->mapFromSource( index ),
246             KDChart::ThreeDLineAttributesRole ) );
247 }
```

7.35.3.100 ThreeDLineAttributes LineDiagram::threeDLineAttributes (int *column*) const

Definition at line 232 of file KDChartLineDiagram.cpp.

References d.

```
{
    return qVariantValue<ThreeDLineAttributes>(
        d->attributesModel->data(
            d->attributesModel->mapFromSource( columnToIndex( column ) ),
            KDChart::ThreeDLineAttributesRole ) );
}
```

7.35.3.101 ThreeDLineAttributes LineDiagram::threeDLineAttributes () const

Definition at line 226 of file KDChartLineDiagram.cpp.

References d.

Referenced by paint(), and threeDItemDepth().

```
{
    return qVariantValue<ThreeDLineAttributes>(
        d->attributesModel->data( KDChart::ThreeDLineAttributesRole ) );
}
```

7.35.3.102 LineDiagram::LineType LineDiagram::type () const

Definition at line 115 of file KDChartLineDiagram.cpp.

References d.

Referenced by calculateDataBoundaries(), and paint().

```
{
   return d->lineType;
}
```

7.35.3.103 void AbstractDiagram::update () const `[inherited]`

Definition at line 961 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractDiagram::doItemsLayout().

7.35.3.104 void KDChart::AbstractDiagram::useDefaultColors () `[inherited]`

Set the palette to be used, for painting datasets to the default palette.

See also:

KDChart::Palette. FIXME: fold into one usePalette (KDChart::Palette&) method

Definition at line 855 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.35.3.105 void KDChart::AbstractDiagram::useRainbowColors () `[inherited]`

Set the palette to be used, for painting datasets to the rainbow palette.

See also:

KDChart::Palette.

Definition at line 865 of file KDChartAbstractDiagram.cpp.

References d.

```
869 {
```

7.35.3.106 bool AbstractDiagram::usesExternalAttributesModel () const `[virtual, inherited]`

Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

See also:

setAttributesModel

Definition at line 280 of file KDChartAbstractDiagram.cpp.

References d.

```
281 {
282     return d->usesExternalAttributesModel();
283 }
```

7.35.3.107 void KDChart::AbstractDiagram::useSubduedColors () `[inherited]`

Set the palette to be used, for painting datasets to the subdued palette.

See also:

KDChart::Palette.

Definition at line 860 of file KDChartAbstractDiagram.cpp.

References d.

```
864 {
```

7.35.3.108 double AbstractDiagram::valueForCell (int *row*, int *column*) const `[protected, inherited]`

Helper method, retrieving the data value (DisplayRole) for a given row and column.

Parameters:

row The row to query.

column The column to query.

Returns:

The value of the display role at the given row and column as a double.

Definition at line 955 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::attributesModelRootIndex(), and d.

Referenced by paint().

```
{
```

7.35.3.109 double LineDiagram::valueForCellTesting (int *row*, int *column*, bool & *bOK*, bool *showHiddenCellsAsInvalid* = false) const `[protected]`

Definition at line 383 of file KDChartLineDiagram.cpp.

References d, and KDChart::AbstractDiagram::isHidden().

Referenced by calculateDataBoundaries(), and getCellValues().

```
{
    double value;
    if( showHiddenCellsAsInvalid && isHidden( model()->index( row, column, rootIndex() ) ) )
        bOK = false;
    else
        value = d->attributesModel->data(
                    d->attributesModel->index( row, column, attributesModelRootIndex() )
                ).toDouble( &bOK );
    return bOK ? value : 0.0;
}
```

7.35.3.110 int AbstractDiagram::verticalOffset () const `[virtual, inherited]`

[reimplemented]

Definition at line 842 of file KDChartAbstractDiagram.cpp.

```
{ return true; }
```

7.35.3.111 QRect AbstractDiagram::visualRect (const QModelIndex & *index*) const `[virtual, inherited]`

[reimplemented]

Definition at line 825 of file KDChartAbstractDiagram.cpp.

```
{}
```

7.35.3.112 QRegion AbstractDiagram::visualRegionForSelection (const QItemSelection & *selection*) const `[virtual, inherited]`

[reimplemented]

Definition at line 851 of file KDChartAbstractDiagram.cpp.

7.35.4 Member Data Documentation

7.35.4.1 Q_SIGNALS KDChart::AbstractDiagram::__pad0__ `[protected, inherited]`

Definition at line 589 of file KDChartAbstractDiagram.h.

The documentation for this class was generated from the following files:

- KDChartLineDiagram.h
- KDChartLineDiagram.cpp

7.36 KDChart::LineLayoutItem Class Reference

`#include <KDChartLayoutItems.h>`

Inheritance diagram for KDChart::LineLayoutItem:Collaboration diagram for KDChart::LineLayoutItem:

Public Member Functions

- virtual Qt::Orientations expandingDirections () const
- virtual QRect geometry () const
- virtual bool isEmpty () const
- LineLayoutItem (AbstractDiagram ∗diagram, int length, const QPen &pen, Qt::Alignment alignment=0)
- virtual QSize maximumSize () const
- virtual QSize minimumSize () const
- virtual void paint (QPainter ∗)
- virtual void paintAll (QPainter &painter)

  Default impl: just call paint.

- virtual void paintCtx (PaintContext ∗context)

  Default impl: Paint the complete item using its layouted position and size.

- QLayout ∗ parentLayout ()
- void removeFromParentLayout ()
- virtual void setGeometry (const QRect &r)
- void setParentLayout (QLayout ∗lay)
- virtual void setParentWidget (QWidget ∗widget)

  Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

- virtual QSize sizeHint () const
- virtual void sizeHintChanged () const

  Report changed size hint: ask the parent widget to recalculate the layout.

Static Public Member Functions

- void paintIntoRect (QPainter ∗painter, const QRect &rect, const QPen &pen)

Protected Attributes

- QWidget ∗ mParent
- QLayout ∗ mParentLayout

7.36.1 Constructor & Destructor Documentation

7.36.1.1 KDChart::LineLayoutItem::LineLayoutItem (AbstractDiagram ∗ *diagram*, int *length*, const QPen & *pen*, Qt::Alignment *alignment* = 0)

Definition at line 612 of file KDChartLayoutItems.cpp.

```
    : AbstractLayoutItem( alignment )
    , mDiagram( diagram )
    , mLength( length )
    , mPen( pen )
{
    //have a mini pen width
    if ( pen.width() < 2 )
        mPen.setWidth( 2 );
}
```

7.36.2 Member Function Documentation

7.36.2.1 Qt::Orientations KDChart::LineLayoutItem::expandingDirections () const `[virtual]`

Definition at line 626 of file KDChartLayoutItems.cpp.

```
{
    return 0; // Grow neither vertically nor horizontally
}
```

7.36.2.2 QRect KDChart::LineLayoutItem::geometry () const `[virtual]`

Definition at line 631 of file KDChartLayoutItems.cpp.

```
{
    return mRect;
}
```

7.36.2.3 bool KDChart::LineLayoutItem::isEmpty () const `[virtual]`

Definition at line 636 of file KDChartLayoutItems.cpp.

```
{
    return false; // never empty, otherwise the layout item would not exist
}
```

7.36.2.4 QSize KDChart::LineLayoutItem::maximumSize () const `[virtual]`

Definition at line 641 of file KDChartLayoutItems.cpp.

References sizeHint().

```
{
    return sizeHint(); // PENDING(kalle) Review, quite inflexible
}
```

7.36.2.5 QSize KDChart::LineLayoutItem::minimumSize () const `[virtual]`

Definition at line 646 of file KDChartLayoutItems.cpp.

References sizeHint().

```
647 {
648     return sizeHint(); // PENDING(kalle) Review, quite inflexible
649 }
```

7.36.2.6 void KDChart::LineLayoutItem::paint (QPainter ∗) `[virtual]`

Implements KDChart::AbstractLayoutItem.

Definition at line 661 of file KDChartLayoutItems.cpp.

References paintIntoRect().

```
662 {
663     paintIntoRect( painter, mRect, mPen );
664 }
```

7.36.2.7 void KDChart::AbstractLayoutItem::paintAll (QPainter & *painter*) `[virtual, inherited]`

Default impl: just call paint.

Derived classes like KDChart::AbstractArea are providing additional action here.

Reimplemented in KDChart::AbstractArea, and KDChart::TextArea.

Definition at line 69 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint().

```
70 {
71     paint( &painter );
72 }
```

7.36.2.8 void KDChart::AbstractLayoutItem::paintCtx (PaintContext ∗ *context*) `[virtual, inherited]`

Default impl: Paint the complete item using its layouted position and size.

Reimplemented in KDChart::CartesianAxis.

Definition at line 77 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint(), and KDChart::PaintContext::painter().

```
78 {
79     if( context )
80         paint( context->painter() );
81 }
```

7.36.2.9 void KDChart::LineLayoutItem::paintIntoRect (QPainter ∗ *painter*, const QRect & *rect*, const QPen & *pen*) `[static]`

Definition at line 666 of file KDChartLayoutItems.cpp.

Referenced by paint().

```
670 {
671     if( ! rect.isValid() )
672         return;
673 
674     const QPen oldPen = painter->pen();
675     painter->setPen( pen );
676     const qreal y = rect.center().y();
677     painter->drawLine( QPointF( rect.left(), y ),
678                        QPointF( rect.right(), y ) );
679     painter->setPen( oldPen );
680 }
```

7.36.2.10 QLayout∗ KDChart::AbstractLayoutItem::parentLayout () `[inherited]`

Definition at line 74 of file KDChartLayoutItems.h.

```
75          {
76              return mParentLayout;
77          }
```

7.36.2.11 void KDChart::AbstractLayoutItem::removeFromParentLayout () `[inherited]`

Definition at line 78 of file KDChartLayoutItems.h.

Referenced by KDChart::Chart::takeCoordinatePlane().

```
79          {
80              if( mParentLayout ){
81                  if( widget() )
82                      mParentLayout->removeWidget( widget() );
83                  else
84                      mParentLayout->removeItem( this );
85              }
86          }
```

7.36.2.12 void KDChart::LineLayoutItem::setGeometry (const QRect & *r*) `[virtual]`

Definition at line 651 of file KDChartLayoutItems.cpp.

```
652 {
653     mRect = r;
654 }
```

7.36.2.13 void KDChart::AbstractLayoutItem::setParentLayout (QLayout ∗ *lay*) `[inherited]`

Definition at line 70 of file KDChartLayoutItems.h.

```
71         {
72             mParentLayout = lay;
73         }
```

7.36.2.14 void KDChart::AbstractLayoutItem::setParentWidget (QWidget ∗ *widget*) `[virtual, inherited]`

Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

Thus, you need to call setParentWidget on every item, that has a non-fixed size.

Definition at line 64 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::mParent.

Referenced by KDChart::Legend::buildLegend(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
65 {
66     mParent = widget;
67 }
```

7.36.2.15 QSize KDChart::LineLayoutItem::sizeHint () const `[virtual]`

Definition at line 656 of file KDChartLayoutItems.cpp.

Referenced by maximumSize(), and minimumSize().

```
657 {
658     return QSize( mLength, mPen.width()+2 );
659 }
```

7.36.2.16 void KDChart::AbstractLayoutItem::sizeHintChanged () const `[virtual, inherited]`

Report changed size hint: ask the parent widget to recalculate the layout.

Definition at line 86 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextLayoutItem::sizeHint().

```
87 {
88     // This is exactly like what QWidget::updateGeometry does.
89 //  qDebug("KDChart::AbstractLayoutItem::sizeHintChanged() called");
90     if( mParent ) {
91         if ( mParent->layout() )
92             mParent->layout()->invalidate();
93         else
94             QApplication::postEvent( mParent, new QEvent( QEvent::LayoutRequest ) );
95     }
96 }
```

7.36.3 Member Data Documentation

7.36.3.1 QWidget* KDChart::AbstractLayoutItem::mParent `[protected, inherited]`

Definition at line 88 of file KDChartLayoutItems.h.

Referenced by KDChart::AbstractLayoutItem::setParentWidget().

7.36.3.2 QLayout* KDChart::AbstractLayoutItem::mParentLayout `[protected, inherited]`

Definition at line 89 of file KDChartLayoutItems.h.

The documentation for this class was generated from the following files:

- KDChartLayoutItems.h
- KDChartLayoutItems.cpp

7.37 KDChart::LineWithMarkerLayoutItem Class Reference

`#include <KDChartLayoutItems.h>`

Inheritance diagram for KDChart::LineWithMarkerLayoutItem:Collaboration diagram for KDChart::LineWithMarkerLayoutItem:

Public Member Functions

- virtual Qt::Orientations expandingDirections () const
- virtual QRect geometry () const
- virtual bool isEmpty () const
- LineWithMarkerLayoutItem (AbstractDiagram ∗diagram, int lineLength, const QPen &linePen, int markerOffs, const MarkerAttributes &marker, const QBrush &markerBrush, const QPen &markerPen, Qt::Alignment alignment=0)
- virtual QSize maximumSize () const
- virtual QSize minimumSize () const
- virtual void paint (QPainter ∗)
- virtual void paintAll (QPainter &painter)

  Default impl: just call paint.

- virtual void paintCtx (PaintContext ∗context)

  Default impl: Paint the complete item using its layouted position and size.

- QLayout ∗ parentLayout ()
- void removeFromParentLayout ()
- virtual void setGeometry (const QRect &r)
- void setParentLayout (QLayout ∗lay)
- virtual void setParentWidget (QWidget ∗widget)

  Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

- virtual QSize sizeHint () const
- virtual void sizeHintChanged () const

  Report changed size hint: ask the parent widget to recalculate the layout.

Protected Attributes

- QWidget ∗ mParent
- QLayout ∗ mParentLayout

7.37.1 Constructor & Destructor Documentation

7.37.1.1 KDChart::LineWithMarkerLayoutItem::LineWithMarkerLayoutItem (AbstractDiagram ∗ *diagram*, int *lineLength*, const QPen & *linePen*, int *markerOffs*, const MarkerAttributes & *marker*, const QBrush & *markerBrush*, const QPen & *markerPen*, Qt::Alignment *alignment* = 0)

Definition at line 683 of file KDChartLayoutItems.cpp.

```
692     : AbstractLayoutItem( alignment )
693     , mDiagram(     diagram )
694     , mLineLength(  lineLength )
695     , mLinePen(     linePen )
696     , mMarkerOffs(  markerOffs )
697     , mMarker(      marker )
698     , mMarkerBrush( markerBrush )
699     , mMarkerPen(   markerPen )
700 {
701 }
```

7.37.2 Member Function Documentation

7.37.2.1 Qt::Orientations KDChart::LineWithMarkerLayoutItem::expandingDirections () const `[virtual]`

Definition at line 703 of file KDChartLayoutItems.cpp.

```
704 {
705     return 0; // Grow neither vertically nor horizontally
706 }
```

7.37.2.2 QRect KDChart::LineWithMarkerLayoutItem::geometry () const `[virtual]`

Definition at line 708 of file KDChartLayoutItems.cpp.

```
709 {
710     return mRect;
711 }
```

7.37.2.3 bool KDChart::LineWithMarkerLayoutItem::isEmpty () const `[virtual]`

Definition at line 713 of file KDChartLayoutItems.cpp.

```
714 {
715     return false; // never empty, otherwise the layout item would not exist
716 }
```

7.37.2.4 QSize KDChart::LineWithMarkerLayoutItem::maximumSize () const `[virtual]`

Definition at line 718 of file KDChartLayoutItems.cpp.

References sizeHint().

```
719 {
720     return sizeHint(); // PENDING(kalle) Review, quite inflexible
721 }
```

7.37.2.5 QSize KDChart::LineWithMarkerLayoutItem::minimumSize () const `[virtual]`

Definition at line 723 of file KDChartLayoutItems.cpp.

References sizeHint().

```
{
    return sizeHint(); // PENDING(kalle) Review, quite inflexible
}
```

7.37.2.6 void KDChart::LineWithMarkerLayoutItem::paint (QPainter ∗) `[virtual]`

Implements KDChart::AbstractLayoutItem.

Definition at line 741 of file KDChartLayoutItems.cpp.

References KDChart::MarkerAttributes::markerSize().

```
{
    // paint the line over the full width, into the vertical middle of the rect
    LineLayoutItem::paintIntoRect( painter, mRect, mLinePen );

    // paint the marker with the given offset from the left side of the line
    const QRect r(
            QPoint( mRect.x()+mMarkerOffs, mRect.y() ),
            QSize( mMarker.markerSize().toSize().width(), mRect.height() ) );
    MarkerLayoutItem::paintIntoRect(
            painter, r, mDiagram, mMarker, mMarkerBrush, mMarkerPen );
}
```

7.37.2.7 void KDChart::AbstractLayoutItem::paintAll (QPainter & *painter*) `[virtual, inherited]`

Default impl: just call paint.

Derived classes like KDChart::AbstractArea are providing additional action here.

Reimplemented in KDChart::AbstractArea, and KDChart::TextArea.

Definition at line 69 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint().

```
{
    paint( &painter );
}
```

7.37.2.8 void KDChart::AbstractLayoutItem::paintCtx (PaintContext ∗ *context*) `[virtual, inherited]`

Default impl: Paint the complete item using its layouted position and size.

Reimplemented in KDChart::CartesianAxis.

Definition at line 77 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint(), and KDChart::PaintContext::painter().

```
{
    if( context )
        paint( context->painter() );
}
```

7.37.2.9 QLayout∗ KDChart::AbstractLayoutItem::parentLayout () `[inherited]`

Definition at line 74 of file KDChartLayoutItems.h.

```
        {
            return mParentLayout;
        }
```

7.37.2.10 void KDChart::AbstractLayoutItem::removeFromParentLayout () `[inherited]`

Definition at line 78 of file KDChartLayoutItems.h.

Referenced by KDChart::Chart::takeCoordinatePlane().

```
        {
            if( mParentLayout ){
                if( widget() )
                    mParentLayout->removeWidget( widget() );
                else
                    mParentLayout->removeItem( this );
            }
        }
```

7.37.2.11 void KDChart::LineWithMarkerLayoutItem::setGeometry (const QRect & *r*) `[virtual]`

Definition at line 728 of file KDChartLayoutItems.cpp.

```
{
    mRect = r;
}
```

7.37.2.12 void KDChart::AbstractLayoutItem::setParentLayout (QLayout ∗ *lay*) `[inherited]`

Definition at line 70 of file KDChartLayoutItems.h.

```
        {
            mParentLayout = lay;
        }
```

7.37.2.13 void KDChart::AbstractLayoutItem::setParentWidget (QWidget ∗ *widget*) `[virtual, inherited]`

Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

Thus, you need to call setParentWidget on every item, that has a non-fixed size.

Definition at line 64 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::mParent.

Referenced by KDChart::Legend::buildLegend(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
{
    mParent = widget;
}
```

7.37.2.14 QSize KDChart::LineWithMarkerLayoutItem::sizeHint () const `[virtual]`

Definition at line 733 of file KDChartLayoutItems.cpp.

References KDChart::MarkerAttributes::markerSize().

Referenced by maximumSize(), and minimumSize().

```
{
    const QSize sizeM = mMarker.markerSize().toSize();
    const QSize sizeL = QSize( mLineLength, mLinePen.width()+2 );
    return QSize( qMax(sizeM.width(),  sizeL.width()),
                  qMax(sizeM.height(), sizeL.height()) );
}
```

7.37.2.15 void KDChart::AbstractLayoutItem::sizeHintChanged () const `[virtual, inherited]`

Report changed size hint: ask the parent widget to recalculate the layout.

Definition at line 86 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextLayoutItem::sizeHint().

```
{
    // This is exactly like what QWidget::updateGeometry does.
//  qDebug("KDChart::AbstractLayoutItem::sizeHintChanged() called");
    if( mParent ) {
        if ( mParent->layout() )
            mParent->layout()->invalidate();
        else
            QApplication::postEvent( mParent, new QEvent( QEvent::LayoutRequest ) );
    }
}
```

7.37.3 Member Data Documentation

7.37.3.1 QWidget∗ KDChart::AbstractLayoutItem::mParent `[protected, inherited]`

Definition at line 88 of file KDChartLayoutItems.h.

Referenced by KDChart::AbstractLayoutItem::setParentWidget().

7.37.3.2 QLayout* KDChart::AbstractLayoutItem::mParentLayout `[protected, inherited]`

Definition at line 89 of file KDChartLayoutItems.h.

The documentation for this class was generated from the following files:

- KDChartLayoutItems.h
- KDChartLayoutItems.cpp

7.38 KDChart::MarkerAttributes Class Reference

`#include <KDChartMarkerAttributes.h>`

Public Types

- enum MarkerStyle {
  MarkerCircle = 0,
  MarkerSquare = 1,
  MarkerDiamond = 2,
  Marker1Pixel = 3,
  Marker4Pixels = 4,
  MarkerRing = 5,
  MarkerCross = 6,
  MarkerFastCross = 7 }
- typedef QMap< uint, MarkerStyle > MarkerStylesMap

Public Member Functions

- bool isVisible () const
- MarkerAttributes (const MarkerAttributes &)
- MarkerAttributes ()
- QColor markerColor () const
- QSizeF markerSize () const
- MarkerStyle markerStyle () const
- MarkerStylesMap markerStylesMap () const
- bool operator!= (const MarkerAttributes &) const
- MarkerAttributes & operator= (const MarkerAttributes &)
- bool operator== (const MarkerAttributes &) const
- QPen pen () const
- void setMarkerColor (const QColor &color)
- void setMarkerSize (const QSizeF &size)

  Normally you need to specify a valid QSizeF here, but for Legends you can use the invalid size QSizeF(), to enable automatic marker size calculation:.

- void setMarkerStyle (MarkerStyle style)
- void setMarkerStylesMap (const MarkerStylesMap &map)
- void setPen (const QPen &pen)
- void setVisible (bool visible)
- ~MarkerAttributes ()

7.38.1 Member Typedef Documentation

7.38.1.1 typedef QMap<uint, MarkerStyle> KDChart::MarkerAttributes::MarkerStylesMap

Definition at line 65 of file KDChartMarkerAttributes.h.

7.38.2 Member Enumeration Documentation

7.38.2.1 enum KDChart::MarkerAttributes::MarkerStyle

Enumeration values:

- ***MarkerCircle***
- ***MarkerSquare***
- ***MarkerDiamond***
- ***Marker1Pixel***
- ***Marker4Pixels***
- ***MarkerRing***
- ***MarkerCross***
- ***MarkerFastCross***

Definition at line 53 of file KDChartMarkerAttributes.h.

```
53                          { MarkerCircle  = 0,
54                            MarkerSquare  = 1,
55                            MarkerDiamond = 2,
56                            Marker1Pixel  = 3,
57                            Marker4Pixels = 4,
58                            MarkerRing    = 5,
59                            MarkerCross   = 6,
60                            MarkerFastCross = 7 };
```

7.38.3 Constructor & Destructor Documentation

7.38.3.1 KDChart::MarkerAttributes::MarkerAttributes ()

7.38.3.2 KDChart::MarkerAttributes::MarkerAttributes (const MarkerAttributes &)

7.38.3.3 KDChart::MarkerAttributes::~MarkerAttributes ()

7.38.4 Member Function Documentation

7.38.4.1 bool KDChart::MarkerAttributes::isVisible () const

Referenced by operator<<(), and KDChart::AbstractDiagram::paintMarker().

7.38.4.2 QColor KDChart::MarkerAttributes::markerColor () const

Referenced by operator<<(), and KDChart::AbstractDiagram::paintMarker().

7.38.4.3 QSizeF KDChart::MarkerAttributes::markerSize () const

Referenced by KDChart::LineWithMarkerLayoutItem::paint(), KDChart::MarkerLayoutItem::paintIntoRect(), KDChart::AbstractDiagram::paintMarker(), KDChart::LineWithMarkerLayoutItem::sizeHint(), and KDChart::MarkerLayoutItem::sizeHint().

7.38.4.4 MarkerStyle KDChart::MarkerAttributes::markerStyle () const

Referenced by operator<<(), and KDChart::AbstractDiagram::paintMarker().

7.38.4.5 MarkerStylesMap KDChart::MarkerAttributes::markerStylesMap () const

Referenced by operator<<().

7.38.4.6 bool KDChart::MarkerAttributes::operator!= (const MarkerAttributes &) const

Definition at line 96 of file KDChartMarkerAttributes.h.

References operator==().

```
96 { return !operator==( other ); }
```

7.38.4.7 MarkerAttributes& KDChart::MarkerAttributes::operator= (const MarkerAttributes &)

7.38.4.8 bool KDChart::MarkerAttributes::operator== (const MarkerAttributes &) const

Referenced by operator!=().

7.38.4.9 QPen KDChart::MarkerAttributes::pen () const

Referenced by operator<<(), and KDChart::AbstractDiagram::paintMarker().

7.38.4.10 void KDChart::MarkerAttributes::setMarkerColor (const QColor & *color*)

7.38.4.11 void KDChart::MarkerAttributes::setMarkerSize (const QSizeF & *size*)

Normally you need to specify a valid QSizeF here, but for Legends you can use the invalid size QSizeF(), to enable automatic marker size calculation:.

For Markers shown in a Legend this means the marker size will be equal to the font height used for the labels that are shown next to the markers.

7.38.4.12 void KDChart::MarkerAttributes::setMarkerStyle (MarkerStyle *style*)

7.38.4.13 void KDChart::MarkerAttributes::setMarkerStylesMap (const MarkerStylesMap & *map*)

7.38.4.14 void KDChart::MarkerAttributes::setPen (const QPen & *pen*)

7.38.4.15 void KDChart::MarkerAttributes::setVisible (bool *visible*)

The documentation for this class was generated from the following file:

- KDChartMarkerAttributes.h

7.39 KDChart::MarkerLayoutItem Class Reference

`#include <KDChartLayoutItems.h>`

Inheritance diagram for KDChart::MarkerLayoutItem:Collaboration diagram for KDChart::MarkerLayoutItem:

Public Member Functions

- virtual Qt::Orientations expandingDirections () const
- virtual QRect geometry () const
- virtual bool isEmpty () const
- MarkerLayoutItem (AbstractDiagram ∗diagram, const MarkerAttributes &marker, const QBrush &brush, const QPen &pen, Qt::Alignment alignment=0)
- virtual QSize maximumSize () const
- virtual QSize minimumSize () const
- virtual void paint (QPainter ∗)
- virtual void paintAll (QPainter &painter)

  Default impl: just call paint.

- virtual void paintCtx (PaintContext ∗context)

  Default impl: Paint the complete item using its layouted position and size.

- QLayout ∗ parentLayout ()
- void removeFromParentLayout ()
- virtual void setGeometry (const QRect &r)
- void setParentLayout (QLayout ∗lay)
- virtual void setParentWidget (QWidget ∗widget)

  Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

- virtual QSize sizeHint () const
- virtual void sizeHintChanged () const

  Report changed size hint: ask the parent widget to recalculate the layout.

Static Public Member Functions

- void paintIntoRect (QPainter ∗painter, const QRect &rect, AbstractDiagram ∗diagram, const MarkerAttributes &marker, const QBrush &brush, const QPen &pen)

Protected Attributes

- QWidget ∗ mParent
- QLayout ∗ mParentLayout

7.39.1 Constructor & Destructor Documentation

7.39.1.1 KDChart::MarkerLayoutItem::MarkerLayoutItem (AbstractDiagram ∗ *diagram*, const MarkerAttributes & *marker*, const QBrush & *brush*, const QPen & *pen*, Qt::Alignment *alignment* = 0)

Definition at line 521 of file KDChartLayoutItems.cpp.

```
    : AbstractLayoutItem( alignment )
    , mDiagram( diagram )
    , mMarker( marker )
    , mBrush( brush )
    , mPen( pen )
{
}
```

7.39.2 Member Function Documentation

7.39.2.1 Qt::Orientations KDChart::MarkerLayoutItem::expandingDirections () const `[virtual]`

Definition at line 533 of file KDChartLayoutItems.cpp.

```
{
    return 0; // Grow neither vertically nor horizontally
}
```

7.39.2.2 QRect KDChart::MarkerLayoutItem::geometry () const `[virtual]`

Definition at line 538 of file KDChartLayoutItems.cpp.

```
{
    return mRect;
}
```

7.39.2.3 bool KDChart::MarkerLayoutItem::isEmpty () const `[virtual]`

Definition at line 543 of file KDChartLayoutItems.cpp.

```
{
    return false; // never empty, otherwise the layout item would not exist
}
```

7.39.2.4 QSize KDChart::MarkerLayoutItem::maximumSize () const `[virtual]`

Definition at line 548 of file KDChartLayoutItems.cpp.

References sizeHint().

```
{
    return sizeHint(); // PENDING(kalle) Review, quite inflexible
}
```

7.39.2.5 QSize KDChart::MarkerLayoutItem::minimumSize () const `[virtual]`

Definition at line 553 of file KDChartLayoutItems.cpp.

References sizeHint().

```
554 {
555     return sizeHint(); // PENDING(kalle) Review, quite inflexible
556 }
```

7.39.2.6 void KDChart::MarkerLayoutItem::paint (QPainter ∗) `[virtual]`

Implements KDChart::AbstractLayoutItem.

Definition at line 569 of file KDChartLayoutItems.cpp.

References paintIntoRect().

```
570 {
571     paintIntoRect( painter, mRect, mDiagram, mMarker, mBrush, mPen );
572 }
```

7.39.2.7 void KDChart::AbstractLayoutItem::paintAll (QPainter & *painter*) `[virtual, inherited]`

Default impl: just call paint.

Derived classes like KDChart::AbstractArea are providing additional action here.

Reimplemented in KDChart::AbstractArea, and KDChart::TextArea.

Definition at line 69 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint().

```
70 {
71     paint( &painter );
72 }
```

7.39.2.8 void KDChart::AbstractLayoutItem::paintCtx (PaintContext ∗ *context*) `[virtual, inherited]`

Default impl: Paint the complete item using its layouted position and size.

Reimplemented in KDChart::CartesianAxis.

Definition at line 77 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint(), and KDChart::PaintContext::painter().

```
78 {
79     if( context )
80         paint( context->painter() );
81 }
```

7.39.2.9 void KDChart::MarkerLayoutItem::paintIntoRect (QPainter ∗ *painter*, const QRect & *rect*, AbstractDiagram ∗ *diagram*, const MarkerAttributes & *marker*, const QBrush & *brush*, const QPen & *pen*) `[static]`

Definition at line 574 of file KDChartLayoutItems.cpp.

References KDChart::MarkerAttributes::markerSize(), and KDChart::AbstractDiagram::paintMarker().

Referenced by paint().

```
581 {
582     if( ! rect.isValid() )
583         return;
584 
585     // The layout management may assign a larger rect than what we
586     // wanted. We need to adjust the position.
587     const QSize siz = marker.markerSize().toSize();
588     QPointF pos = rect.topLeft();
589     pos += QPointF( static_cast<qreal>(( rect.width()  - siz.width()) / 2.0 ),
590                     static_cast<qreal>(( rect.height() - siz.height()) / 2.0 ) );
591 
592 #ifdef DEBUG_ITEMS_PAINT
593     QPointF oldPos = pos;
594 #endif
595 
596 // And finally, drawMarker() assumes the position to be the center
597     // of the marker, adjust again.
598     pos += QPointF( static_cast<qreal>( siz.width() ) / 2.0,
599                     static_cast<qreal>( siz.height() )/ 2.0 );
600 
601     diagram->paintMarker( painter, marker, brush, pen, pos.toPoint(), siz );
602 
603 #ifdef DEBUG_ITEMS_PAINT
604     const QPen oldPen( painter->pen() );
605     painter->setPen( Qt::red );
606     painter->drawRect( QRect(oldPos.toPoint(), siz) );
607     painter->setPen( oldPen );
608 #endif
609 }
```

7.39.2.10 QLayout∗ KDChart::AbstractLayoutItem::parentLayout () `[inherited]`

Definition at line 74 of file KDChartLayoutItems.h.

```
75          {
76              return mParentLayout;
77          }
```

7.39.2.11 void KDChart::AbstractLayoutItem::removeFromParentLayout () `[inherited]`

Definition at line 78 of file KDChartLayoutItems.h.

Referenced by KDChart::Chart::takeCoordinatePlane().

```
79          {
80              if( mParentLayout ){
81                  if( widget() )
82                      mParentLayout->removeWidget( widget() );
83                  else
84                      mParentLayout->removeItem( this );
85              }
86          }
```

7.39.2.12 void KDChart::MarkerLayoutItem::setGeometry (const QRect & *r*) `[virtual]`

Definition at line 558 of file KDChartLayoutItems.cpp.

```
559 {
560     mRect = r;
561 }
```

7.39.2.13 void KDChart::AbstractLayoutItem::setParentLayout (QLayout ∗ *lay*) `[inherited]`

Definition at line 70 of file KDChartLayoutItems.h.

```
71          {
72              mParentLayout = lay;
73          }
```

7.39.2.14 void KDChart::AbstractLayoutItem::setParentWidget (QWidget ∗ *widget*) `[virtual, inherited]`

Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

Thus, you need to call setParentWidget on every item, that has a non-fixed size.

Definition at line 64 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::mParent.

Referenced by KDChart::Legend::buildLegend(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
65 {
66     mParent = widget;
67 }
```

7.39.2.15 QSize KDChart::MarkerLayoutItem::sizeHint () const `[virtual]`

Definition at line 563 of file KDChartLayoutItems.cpp.

References KDChart::MarkerAttributes::markerSize().

Referenced by maximumSize(), and minimumSize().

```
564 {
565     //qDebug() << "KDChart::MarkerLayoutItem::sizeHint() returns:"<<mMarker.markerSize().toSize();
566     return mMarker.markerSize().toSize();
567 }
```

7.39.2.16 void KDChart::AbstractLayoutItem::sizeHintChanged () const `[virtual, inherited]`

Report changed size hint: ask the parent widget to recalculate the layout.

Definition at line 86 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextLayoutItem::sizeHint().

```
{
    // This is exactly like what QWidget::updateGeometry does.
//  qDebug("KDChart::AbstractLayoutItem::sizeHintChanged() called");
    if( mParent ) {
        if ( mParent->layout() )
            mParent->layout()->invalidate();
        else
            QApplication::postEvent( mParent, new QEvent( QEvent::LayoutRequest ) );
    }
}
```

7.39.3 Member Data Documentation

7.39.3.1 QWidget∗ KDChart::AbstractLayoutItem::mParent `[protected, inherited]`

Definition at line 88 of file KDChartLayoutItems.h.

Referenced by KDChart::AbstractLayoutItem::setParentWidget().

7.39.3.2 QLayout∗ KDChart::AbstractLayoutItem::mParentLayout `[protected, inherited]`

Definition at line 89 of file KDChartLayoutItems.h.

The documentation for this class was generated from the following files:

- KDChartLayoutItems.h
- KDChartLayoutItems.cpp

7.40 KDChart::Measure Class Reference

`#include <KDChartMeasure>`

Collaboration diagram for KDChart::Measure:

7.40.1 Detailed Description

Measure is used to specify all relative and/or absolute measures in KDChart, e.g. font sizes.

Definition at line 56 of file KDChartMeasure.h.

Public Member Functions

- qreal calculatedValue (const QSizeF &autoSize, KDChartEnums::MeasureOrientation autoOrientation) const
- qreal calculatedValue (const QObject ∗autoArea, KDChartEnums::MeasureOrientation autoOrientation) const

  The reference area must either be derived from AbstractArea or be derived from QWidget, so e.g.

- KDChartEnums::MeasureCalculationMode calculationMode () const
- Measure (const Measure &)
- Measure (qreal value, KDChartEnums::MeasureCalculationMode mode=KDChartEnums::MeasureCalculationModeAuto, KDChartEnums::MeasureOrientation orientation=KDChartEnums::MeasureOrientationAuto)
- Measure ()
- bool operator!= (const Measure &other) const
- Measure & operator= (const Measure &)
- bool operator== (const Measure &) const
- const QObject ∗ referenceArea () const

  The returned reference area will either be derived from AbstractArea or be derived from QWidget.

- KDChartEnums::MeasureOrientation referenceOrientation () const
- void setAbsoluteValue (qreal val)

  This is a convenience method for specifying a value, with implicitely setting the calculation mode to MeasureCalculationModeAbsolute.

- void setCalculationMode (KDChartEnums::MeasureCalculationMode mode)
- void setReferenceArea (const QObject ∗area)

  The reference area must either be derived from AbstractArea or be derived from QWidget, so e.g.

- void setReferenceOrientation (KDChartEnums::MeasureOrientation orientation)
- void setRelativeMode (const QObject ∗area, KDChartEnums::MeasureOrientation orientation)

  The reference area must either be derived from AbstractArea or be derived from QWidget, so e.g.

- void setValue (qreal val)
- const QSizeF sizeOfArea (const QObject ∗area) const
- qreal value () const

7.40.2 Constructor & Destructor Documentation

7.40.2.1 KDChart::Measure::Measure ()

Definition at line 41 of file KDChartMeasure.cpp.

```
42   : mValue( 0.0 ),
43     mMode(  KDChartEnums::MeasureCalculationModeAuto ),
44     mArea(  0 ),
45     mOrientation( KDChartEnums::MeasureOrientationAuto )
46 {
47     // this bloc left empty intentionally
48 }
```

7.40.2.2 KDChart::Measure::Measure (qreal *value*, KDChartEnums::MeasureCalculationMode *mode* = KDChartEnums::MeasureCalculationModeAuto, KDChartEnums::MeasureOrientation *orientation* = KDChartEnums::MeasureOrientationAuto)

Definition at line 50 of file KDChartMeasure.cpp.

```
53   : mValue( value ),
54     mMode(  mode ),
55     mArea(  0 ),
56     mOrientation( orientation )
57 {
58     // this bloc left empty intentionally
59 }
```

7.40.2.3 KDChart::Measure::Measure (const Measure &)

Definition at line 61 of file KDChartMeasure.cpp.

```
62   : mValue( r.value() ),
63     mMode(  r.calculationMode() ),
64     mArea(  r.referenceArea() ),
65     mOrientation( r.referenceOrientation() )
66 {
67     // this bloc left empty intentionally
68 }
```

7.40.3 Member Function Documentation

7.40.3.1 qreal KDChart::Measure::calculatedValue (const QSizeF & *autoSize*, KDChartEnums::MeasureOrientation *autoOrientation*) const

Definition at line 83 of file KDChartMeasure.cpp.

References sizeOfArea(), and value().

```
85 {
86     if( mMode == KDChartEnums::MeasureCalculationModeAbsolute ){
87         return mValue;
88     }else{
89         qreal value = 0.0;
90         const QObject theAutoArea;
```

```
        const QObject* autoArea = &theAutoArea;
        const QObject* area = mArea ? mArea : autoArea;
        KDChartEnums::MeasureOrientation orientation = mOrientation;
        switch( mMode ){
            case KDChartEnums::MeasureCalculationModeAuto:
                area = autoArea;
                orientation = autoOrientation;
                break;
            case KDChartEnums::MeasureCalculationModeAutoArea:
                area = autoArea;
                break;
            case KDChartEnums::MeasureCalculationModeAutoOrientation:
                orientation = autoOrientation;
                break;
            case KDChartEnums::MeasureCalculationModeAbsolute: // fall through intended
            case KDChartEnums::MeasureCalculationModeRelative:
                break;
        }
        if( area ){
            QSizeF size;
            if( area == autoArea )
                size = autoSize;
            else
                size = sizeOfArea( area );
            //qDebug() << "size" << size;
            qreal referenceValue;
            switch( orientation ){
                case KDChartEnums::MeasureOrientationAuto: // fall through intended
                case KDChartEnums::MeasureOrientationMinimum:
                    referenceValue = qMin( size.width(), size.height() );
                    break;
                case KDChartEnums::MeasureOrientationMaximum:
                    referenceValue = qMax( size.width(), size.height() );
                    break;
                case KDChartEnums::MeasureOrientationHorizontal:
                    referenceValue = size.width();
                    break;
                case KDChartEnums::MeasureOrientationVertical:
                    referenceValue = size.height();
                    break;
            }
            value = mValue / 1000.0 * referenceValue;
        }
        return value;
    }
}
```

7.40.3.2 qreal KDChart::Measure::calculatedValue (const QObject ∗ *autoArea*, KDChartEnums::MeasureOrientation *autoOrientation*) const

The reference area must either be derived from AbstractArea or be derived from QWidget, so e.g.

it could be derived from AbstractAreaWidget too.

Definition at line 139 of file KDChartMeasure.cpp.

References sizeOfArea().

```
{
    return calculatedValue( sizeOfArea( autoArea ), autoOrientation);
}
```

7.40.3.3 KDChartEnums::MeasureCalculationMode KDChart::Measure::calculationMode () const

Definition at line 70 of file KDChartMeasure.h.

Referenced by operator<<(), operator=(), and operator==().

```
70 { return mMode; }
```

7.40.3.4 bool KDChart::Measure::operator!= (const Measure & *other*) const

Definition at line 126 of file KDChartMeasure.h.

```
126 { return !operator==(other); }
```

7.40.3.5 Measure & KDChart::Measure::operator= (const Measure &)

Definition at line 70 of file KDChartMeasure.cpp.

References calculationMode(), referenceArea(), referenceOrientation(), and value().

```
71 {
72     if( this != &r ){
73         mValue = r.value();
74         mMode  = r.calculationMode();
75         mArea  = r.referenceArea();
76         mOrientation = r.referenceOrientation();
77     }
78
79     return *this;
80 }
```

7.40.3.6 bool KDChart::Measure::operator== (const Measure &) const

Definition at line 177 of file KDChartMeasure.cpp.

References calculationMode(), referenceArea(), referenceOrientation(), and value().

```
178 {
179     return( mValue == r.value() &&
180             mMode  == r.calculationMode() &&
181             mArea  == r.referenceArea() &&
182             mOrientation == r.referenceOrientation() );
183 }
```

7.40.3.7 const QObject∗ KDChart::Measure::referenceArea () const

The returned reference area will either be derived from AbstractArea or be derived from QWidget.

Definition at line 111 of file KDChartMeasure.h.

Referenced by operator<<(), operator=(), and operator==().

```
111 { return mArea; }
```

7.40.3.8 KDChartEnums::MeasureOrientation KDChart::Measure::referenceOrientation () const

Definition at line 114 of file KDChartMeasure.h.

Referenced by operator<<(), operator=(), and operator==().

```
114 { return mOrientation; }
```

7.40.3.9 void KDChart::Measure::setAbsoluteValue (qreal *val*)

This is a convenience method for specifying a value, with implicitely setting the calculation mode to MeasureCalculationModeAbsolute.

Calling setAbsoluteValue(value) is the same as calling

```
    setValue( value );
    setCalculationMode( KDChartEnums::MeasureCalculationModeAbsolute );
```

Definition at line 95 of file KDChartMeasure.h.

```
96      {
97          mMode = KDChartEnums::MeasureCalculationModeAbsolute;
98          mValue = val;
99      }
```

7.40.3.10 void KDChart::Measure::setCalculationMode (KDChartEnums::MeasureCalculationMode *mode*)

Definition at line 69 of file KDChartMeasure.h.

```
69 { mMode = mode; }
```

7.40.3.11 void KDChart::Measure::setReferenceArea (const QObject ∗ *area*)

The reference area must either be derived from AbstractArea or be derived from QWidget, so e.g.

it could be derived from AbstractAreaWidget too.

Definition at line 106 of file KDChartMeasure.h.

```
106 { mArea = area; }
```

7.40.3.12 void KDChart::Measure::setReferenceOrientation (KDChartEnums::MeasureOrientation *orientation*)

Definition at line 113 of file KDChartMeasure.h.

```
113 { mOrientation = orientation; }
```

7.40.3.13 void KDChart::Measure::setRelativeMode (const QObject ∗ *area*, KDChartEnums::MeasureOrientation *orientation*)

The reference area must either be derived from AbstractArea or be derived from QWidget, so e.g. it could be derived from AbstractAreaWidget too.

Definition at line 77 of file KDChartMeasure.h.

Referenced by KDChart::Chart::addLegend().

```
79     {
80         mMode = KDChartEnums::MeasureCalculationModeRelative;
81         mArea = area;
82         mOrientation = orientation;
83     }
```

7.40.3.14 void KDChart::Measure::setValue (qreal *val*)

Definition at line 66 of file KDChartMeasure.h.

Referenced by KDChart::Chart::addLegend(), and KDChart::CartesianAxis::titleTextAttributes().

```
66 { mValue = val; }
```

7.40.3.15 const QSizeF KDChart::Measure::sizeOfArea (const QObject ∗ *area*) const

Definition at line 146 of file KDChartMeasure.cpp.

Referenced by calculatedValue().

```
147 {
148     QSizeF size;
149     const AbstractArea* kdcArea = dynamic_cast<const AbstractArea*>(area);
150     if( kdcArea ){
151         size = kdcArea->geometry().size();
152         //qDebug() << "Measure::sizeOfArea() found kdcArea with size" << size;
153     }else{
154         const QWidget* widget = dynamic_cast<const QWidget*>(area);
155         if( widget ){
156             /* ATTENTION: Using the layout does not work: The Legend will never get the right size the
157             const QLayout * layout = widget->layout();
158             if( layout ){
159                 size = layout->geometry().size();
160                 //qDebug() << "Measure::sizeOfArea() found widget with layout size" << size;
161             }else*/
162             {
163                 size = widget->geometry().size();
164                 //qDebug() << "Measure::sizeOfArea() found widget with size" << size;
165             }
166         }else if( mMode != KDChartEnums::MeasureCalculationModeAbsolute ){
167             size = QSizeF(1.0, 1.0);
168             //qDebug("Measure::sizeOfArea() got no valid area.");
169         }
170     }
171     const QPair< qreal, qreal > factors
172             = GlobalMeasureScaling::instance()->currentFactors();
173     return QSizeF(size.width() * factors.first, size.height() * factors.second);
174 }
```

7.40.3.16 qreal KDChart::Measure::value () const

Definition at line 67 of file KDChartMeasure.h.

Referenced by calculatedValue(), operator<<(), operator=(), operator==(), and KDChart::CartesianAxis::titleTextAttributes().

```
67 { return mValue; }
```

The documentation for this class was generated from the following files:

- KDChartMeasure.h
- KDChartMeasure.cpp

7.41 KDChart::PaintContext Class Reference

`#include <KDChartPaintContext.h>`

Collaboration diagram for KDChart::PaintContext:

Public Member Functions

- AbstractCoordinatePlane * coordinatePlane () const
- PaintContext ()
- QPainter * painter () const
- const QRectF rectangle () const
- void setCoordinatePlane (AbstractCoordinatePlane *plane)
- void setPainter (QPainter *painter)
- void setRectangle (const QRectF &rect)
- ∼PaintContext ()

7.41.1 Constructor & Destructor Documentation

7.41.1.1 KDChart::PaintContext::PaintContext ()

7.41.1.2 KDChart::PaintContext::∼PaintContext ()

7.41.2 Member Function Documentation

7.41.2.1 AbstractCoordinatePlane* KDChart::PaintContext::coordinatePlane () const

Referenced by KDChart::CartesianAxis::paintCtx().

7.41.2.2 QPainter* KDChart::PaintContext::painter () const

Referenced by KDChart::RingDiagram::paint(), KDChart::PolarDiagram::paint(), KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::AbstractLayoutItem::paintCtx(), KDChart::CartesianAxis::paintCtx(), and KDChart::PolarDiagram::paintPolarMarkers().

7.41.2.3 const QRectF KDChart::PaintContext::rectangle () const

Referenced by KDChart::PolarDiagram::paint(), KDChart::PieDiagram::paint(), and KDChart::BarDiagram::paint().

7.41.2.4 void KDChart::PaintContext::setCoordinatePlane (AbstractCoordinatePlane * *plane*)

Referenced by KDChart::PolarCoordinatePlane::paint(), KDChart::CartesianCoordinatePlane::paint(), and KDChart::CartesianAxis::paint().

7.41.2.5 void KDChart::PaintContext::setPainter (QPainter ∗ *painter*)

Referenced by KDChart::PolarCoordinatePlane::paint(), KDChart::CartesianCoordinatePlane::paint(), KDChart::CartesianAxis::paint(), KDChart::RingDiagram::paintEvent(), KDChart::PolarDiagram::paintEvent(), KDChart::PieDiagram::paintEvent(), and KDChart::LineDiagram::paintEvent().

7.41.2.6 void KDChart::PaintContext::setRectangle (const QRectF & *rect*)

Referenced by KDChart::PolarCoordinatePlane::paint(), KDChart::CartesianCoordinatePlane::paint(), KDChart::CartesianAxis::paint(), KDChart::RingDiagram::paintEvent(), KDChart::PolarDiagram::paintEvent(), KDChart::PieDiagram::paintEvent(), and KDChart::LineDiagram::paintEvent().

The documentation for this class was generated from the following file:

- KDChartPaintContext.h

7.42 KDChart::Palette Class Reference

`#include <KDChartPalette.h>`

Inheritance diagram for KDChart::Palette:Collaboration diagram for KDChart::Palette:

7.42.1 Detailed Description

A Palette is a set of brushes (or colors) to be used for painting data sets.

The palette class encapsulates a colletion of brushes, which in the simplest case are colors, to be used for painting a series of data sets. When asked for the m-th color, a palette of size n will wrap around and thus cycle through the available colors.

Three builtin palettes are provided for convenience, one with a default set of colors, one with a subdued color selection, one with rainbow colors.

When a palette changes, it emits a changed() signal. Hook up to it, if you want to repaint when the color selection changes.

Definition at line 55 of file KDChartPalette.h.

Public Member Functions

- void addBrush (const QBrush &brush, int position=-1)

  Adds brush *to the palette.*

- QBrush getBrush (int position) const

  Query the palette for a brush at the specified position.

- bool isValid () const

  Returns wether this represents a valid palette.

- Palette & operator= (const Palette &)
- Palette (const Palette &)
- Palette (QObject ∗parent=0)
- void removeBrush (int position)

  Remove the brush at position.

- int size () const

  Return the number of brushed in the palette.

- ∼Palette ()

Static Public Member Functions

- const Palette & defaultPalette ()

  Provide access to the three builtin palettes, one with standard bright colors, one with more subdued colors, and one with rainbow colors.

- const Palette & rainbowPalette ()
- const Palette & subduedPalette ()

Public Attributes

- Q_SIGNALS __pad0__: void changed()

7.42.2 Constructor & Destructor Documentation

7.42.2.1 KDChart::Palette::Palette (QObject ∗ *parent* = 0) `[explicit]`

7.42.2.2 KDChart::Palette::Palette (const Palette &)

7.42.2.3 KDChart::Palette::∼Palette ()

7.42.3 Member Function Documentation

7.42.3.1 void KDChart::Palette::addBrush (const QBrush & *brush*, int *position* = -1)

Adds *brush* to the palette.

If no position is specified, the brush is appended.

Referenced by makeDefaultPalette(), makeRainbowPalette(), and makeSubduedPalette().

7.42.3.2 const Palette& KDChart::Palette::defaultPalette () `[static]`

Provide access to the three builtin palettes, one with standard bright colors, one with more subdued colors, and one with rainbow colors.

Referenced by KDChart::AttributesModel::headerData().

7.42.3.3 QBrush KDChart::Palette::getBrush (int *position*) const

Query the palette for a brush at the specified position.

If the position exceeds the size of the palette, it wraps around.

Referenced by KDChart::AttributesModel::headerData(), and makeRainbowPalette().

7.42.3.4 bool KDChart::Palette::isValid () const

Returns wether this represents a valid palette.

For a palette to be valid it needs to have at least one brush associated.

7.42.3.5 Palette& KDChart::Palette::operator= (const Palette &)

7.42.3.6 const Palette& KDChart::Palette::rainbowPalette () `[static]`

Referenced by KDChart::AttributesModel::headerData().

7.42.3.7 void KDChart::Palette::removeBrush (int *position*)

Remove the brush at position.

Parameters:

position,if there is one.

7.42.3.8 int KDChart::Palette::size () const

Return the number of brushed in the palette.

7.42.3.9 const Palette& KDChart::Palette::subduedPalette () `[static]`

Referenced by KDChart::AttributesModel::headerData().

7.42.4 Member Data Documentation

7.42.4.1 Q_SIGNALS KDChart::Palette::__pad0__

Definition at line 94 of file KDChartPalette.h.

The documentation for this class was generated from the following file:

- KDChartPalette.h

7.43 KDChart::PieAttributes Class Reference

`#include <KDChartPieAttributes.h>`

Public Member Functions

- bool explode () const
- qreal explodeFactor () const
- bool operator!= (const PieAttributes &other) const
- PieAttributes & operator= (const PieAttributes &)
- bool operator== (const PieAttributes &) const
- PieAttributes (const PieAttributes &)
- PieAttributes ()
- void setExplode (bool explode)

  Enable or disable exploding the respective pie piece(s).
- void setExplodeFactor (qreal factor)

  Set the explode factor.
- ∼PieAttributes ()

7.43.1 Constructor & Destructor Documentation

7.43.1.1 PieAttributes::PieAttributes ()

Definition at line 45 of file KDChartPieAttributes.cpp.

```
46     : _d( new Private() )
47 {
48 }
```

7.43.1.2 PieAttributes::PieAttributes (const PieAttributes &)

Definition at line 50 of file KDChartPieAttributes.cpp.

References d.

```
51     : _d( new Private( *r.d ) )
52 {
53 }
```

7.43.1.3 PieAttributes::∼PieAttributes ()

Definition at line 65 of file KDChartPieAttributes.cpp.

```
66 {
67     delete _d; _d = 0;
68 }
```

7.43.2 Member Function Documentation

7.43.2.1 bool PieAttributes::explode () const

Returns:

whether the respective pie piece(s) will be exploded.

Definition at line 90 of file KDChartPieAttributes.cpp.

References d.

Referenced by KDChart::PieDiagram::calculateDataBoundaries().

```
91 {
92     return (d->explodeFactor != 0.0);
93 }
```

7.43.2.2 qreal PieAttributes::explodeFactor () const

Returns:

the explode factor set by setExplode or by setExplodeFactor.

Definition at line 100 of file KDChartPieAttributes.cpp.

References d.

Referenced by KDChart::PieDiagram::calculateDataBoundaries(), operator<<(), operator==(), and KDChart::PieDiagram::paint().

```
101 {
102     return d->explodeFactor;
103 }
```

7.43.2.3 bool KDChart::PieAttributes::operator!= (const PieAttributes & *other*) const

Definition at line 72 of file KDChartPieAttributes.h.

```
72 { return !operator==(other); }
```

7.43.2.4 PieAttributes & PieAttributes::operator= (const PieAttributes &)

Definition at line 55 of file KDChartPieAttributes.cpp.

References d.

```
56 {
57     if( this == &r )
58         return *this;
59
60     *d = *r.d;
61
62     return *this;
63 }
```

7.43.2.5 bool PieAttributes::operator== (const PieAttributes &) const

Definition at line 71 of file KDChartPieAttributes.cpp.

References explodeFactor().

```
{
    if( explodeFactor() == r.explodeFactor() )
        return true;
    else
        return false;
}
```

7.43.2.6 void PieAttributes::setExplode (bool *explode*)

Enable or disable exploding the respective pie piece(s).

The default explode factor is 10 percent; use setExplodeFactor to specify a different factor.

Note:

This is a convenience function: Calling setExplode(true) does the same as calling setExplodeFactor(0.1), and calling setExplode(false) does the same as calling setExplodeFactor(0.0).

See also:

setExplodeFactor

Definition at line 85 of file KDChartPieAttributes.cpp.

References d.

```
{
    d->explodeFactor = (enabled ? 0.1 : 0.0);
}
```

7.43.2.7 void PieAttributes::setExplodeFactor (qreal *factor*)

Set the explode factor.

The explode factor is a qreal between 0 and 1, and is interpreted as a percentage of the total available radius of the pie.

See also:

setExplode

Definition at line 95 of file KDChartPieAttributes.cpp.

References d.

```
{
    d->explodeFactor = factor;
}
```

The documentation for this class was generated from the following files:

- KDChartPieAttributes.h
- KDChartPieAttributes.cpp

7.44 KDChart::PieDiagram Class Reference

`#include <KDChartPieDiagram.h>`

Inheritance diagram for KDChart::PieDiagram:Collaboration diagram for KDChart::PieDiagram:

Public Member Functions

- bool allowOverlappingDataValueTexts () const
- bool antiAliasing () const
- virtual AttributesModel ∗ attributesModel () const

  Returns the AttributesModel, that is used by this diagram.

- QBrush brush (const QModelIndex &index) const

  Retrieve the brush to be used, for painting the datapoint at the given index in the model.

- QBrush brush (int dataset) const

  Retrieve the brush to be used for the given dataset.

- QBrush brush () const

  Retrieve the brush to be used for painting datapoints globally.

- virtual PieDiagram ∗ clone () const
- int columnCount () const
- bool compare (const AbstractDiagram ∗other) const

  Returns true if both diagrams have the same settings.

- AbstractCoordinatePlane ∗ coordinatePlane () const

  The coordinate plane associated with the diagram.

- const QPair< QPointF, QPointF > dataBoundaries () const

  Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

- virtual void dataChanged (const QModelIndex &topLeft, const QModelIndex &bottomRight)

  [reimplemented]

- QList< QBrush > datasetBrushes () const

  The set of dataset brushes currently used, for use in legends, etc.

- int datasetDimension () const

  The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

- QStringList datasetLabels () const

  The set of dataset labels currently displayed, for use in legends, etc.

- QList< MarkerAttributes > datasetMarkers () const

  The set of dataset markers currently used, for use in legends, etc.

- QList< QPen > datasetPens () const

  The set of dataset pens currently used, for use in legends, etc.

- DataValueAttributes dataValueAttributes (const QModelIndex &index) const

  Retrieve the DataValueAttributes for the given index.

- DataValueAttributes dataValueAttributes (int column) const

  Retrieve the DataValueAttributes for the given dataset.

- DataValueAttributes dataValueAttributes () const

  Retrieve the DataValueAttributes speficied globally.

- virtual void doItemsLayout ()

  [reimplemented]

- qreal granularity () const
- virtual int horizontalOffset () const

  [reimplemented]

- virtual QModelIndex indexAt (const QPoint &point) const

  [reimplemented]

- bool isHidden (const QModelIndex &index) const

  Retrieve the hidden status for the given index.

- bool isHidden (int column) const

  Retrieve the hidden status for the given dataset.

- bool isHidden () const

  Retrieve the hidden status speficied globally.

- virtual bool isIndexHidden (const QModelIndex &index) const

  [reimplemented]

- QStringList itemRowLabels () const

  The set of item row labels currently displayed, for use in Abscissa axes, etc.

- virtual QModelIndex moveCursor (CursorAction cursorAction, Qt::KeyboardModifiers modifiers)

  [reimplemented]

- virtual double numberOfGridRings () const

  [reimplemented]

- virtual double numberOfValuesPerDataset () const

  [reimplemented]

- void paintDataValueText (QPainter ∗painter, const QModelIndex &index, const QPointF &pos, double value)
- void paintMarker (QPainter ∗painter, const QModelIndex &index, const QPointF &pos)

- virtual void paintMarker (QPainter ∗painter, const MarkerAttributes &markerAttributes, const QBrush &brush, const QPen &, const QPointF &point, const QSizeF &size)
- QPen pen (const QModelIndex &index) const

  Retrieve the pen to be used, for painting the datapoint at the given index in the model.
- QPen pen (int dataset) const

  Retrieve the pen to be used for the given dataset.
- QPen pen () const

  Retrieve the pen to be used for painting datapoints globally.
- bool percentMode () const
- PieAttributes pieAttributes (const QModelIndex &index) const
- PieAttributes pieAttributes (int column) const
- PieAttributes pieAttributes () const
- PieDiagram (QWidget ∗parent=0, PolarCoordinatePlane ∗plane=0)
- const PolarCoordinatePlane ∗ polarCoordinatePlane () const
- virtual void resize (const QSizeF &area)

  [reimplemented]
- virtual void scrollTo (const QModelIndex &index, ScrollHint hint=EnsureVisible)

  [reimplemented]
- void setAllowOverlappingDataValueTexts (bool allow)

  Set whether data value labels are allowed to overlap.
- void setAntiAliasing (bool enabled)

  Set whether anti-aliasing is to be used while rendering this diagram.
- virtual void setAttributesModel (AttributesModel ∗model)

  Associate an AttributesModel with this diagram.
- void setBrush (const QBrush &brush)

  Set the brush to be used, for painting all datasets in the model.
- void setBrush (int dataset, const QBrush &brush)

  Set the brush to be used, for painting the given dataset.
- void setBrush (const QModelIndex &index, const QBrush &brush)

  Set the brush to be used, for painting the datapoint at the given index.
- virtual void setCoordinatePlane (AbstractCoordinatePlane ∗plane)

  Set the coordinate plane associated with the diagram.
- void setDatasetDimension (int dimension)

  Sets the dataset dimension of the diagram.
- void setDataValueAttributes (const DataValueAttributes &a)

  Set the DataValueAttributes for all datapoints in the model.

- void setDataValueAttributes (int dataset, const DataValueAttributes &a)

  Set the DataValueAttributes for the given dataset.

- void setDataValueAttributes (const QModelIndex &index, const DataValueAttributes &a)

  Set the DataValueAttributes for the given index.

- void setGranularity (qreal value)

  Set the granularity: the smaller the granularity the more your diagram segments will show facettes instead of rounded segments.

- void setHidden (bool hidden)

  Hide (or unhide, resp.) all datapoints in the model.

- void setHidden (int column, bool hidden)

  Hide (or unhide, resp.) a dataset.

- void setHidden (const QModelIndex &index, bool hidden)

  Hide (or unhide, resp.) a data cell.

- virtual void setModel (QAbstractItemModel ∗model)

  Associate a model with the diagram.

- void setPen (const QPen &pen)

  Set the pen to be used, for painting all datasets in the model.

- void setPen (int dataset, const QPen &pen)

  Set the pen to be used, for painting the given dataset.

- void setPen (const QModelIndex &index, const QPen &pen)

  Set the pen to be used, for painting the datapoint at the given index.

- void setPercentMode (bool percent)
- void setPieAttributes (int column, const PieAttributes &a)
- void setPieAttributes (const PieAttributes &a)
- virtual void setRootIndex (const QModelIndex &idx)

  Set the root index in the model, where the diagram starts referencing data for display.

- virtual void setSelection (const QRect &rect, QItemSelectionModel::SelectionFlags command)

  [reimplemented]

- void setStartPosition (int degrees)
- void setThreeDPieAttributes (const QModelIndex &index, const ThreeDPieAttributes &a)
- void setThreeDPieAttributes (int column, const ThreeDPieAttributes &a)
- void setThreeDPieAttributes (const ThreeDPieAttributes &a)
- int startPosition () const
- ThreeDPieAttributes threeDPieAttributes (const QModelIndex &index) const
- ThreeDPieAttributes threeDPieAttributes (int column) const
- ThreeDPieAttributes threeDPieAttributes () const
- void update () const
- void useDefaultColors ()

Set the palette to be used, for painting datasets to the default palette.

- void useRainbowColors ()

  Set the palette to be used, for painting datasets to the rainbow palette.

- virtual bool usesExternalAttributesModel () const

  Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

- void useSubduedColors ()

  Set the palette to be used, for painting datasets to the subdued palette.

- virtual double valueTotals () const

  [reimplemented]

- virtual int verticalOffset () const

  [reimplemented]

- virtual QRect visualRect (const QModelIndex &index) const

  [reimplemented]

- virtual QRegion visualRegionForSelection (const QItemSelection &selection) const

  [reimplemented]

- virtual ∼PieDiagram ()

Protected Member Functions

- QModelIndex attributesModelRootIndex () const
- virtual const QPair< QPointF, QPointF > calculateDataBoundaries () const

  [reimplemented]

- virtual bool checkInvariants (bool justReturnTheStatus=false) const
- QModelIndex columnToIndex (int column) const
- void dataHidden ()

  This signal is emitted, when the hidden status of at least one data cell was (un)set.

- void modelsChanged ()

  This signal is emitted, when either the model or the AttributesModel is replaced.

- virtual void paint (PaintContext ∗paintContext)

  [reimplemented]

- virtual void paintDataValueTexts (QPainter ∗painter)
- void paintEvent (QPaintEvent ∗)
- virtual void paintMarkers (QPainter ∗painter)
- void propertiesChanged ()

  Emitted upon change of a property of the Diagram.

- void resizeEvent (QResizeEvent ∗)

- void setAttributesModelRootIndex (const QModelIndex &)
- void setDataBoundariesDirty () const
- double valueForCell (int row, int column) const

  Helper method, retrieving the data value (DisplayRole) for a given row and column.

Protected Attributes

- Q_SIGNALS __pad0__: void layoutChanged(AbstractDiagram∗)

7.44.1 Constructor & Destructor Documentation

7.44.1.1 PieDiagram::PieDiagram (QWidget ∗ *parent* = 0, PolarCoordinatePlane ∗ *plane* = 0) `[explicit]`

Definition at line 52 of file KDChartPieDiagram.cpp.

Referenced by clone().

```
52                                                                       :
53      AbstractPieDiagram( new Private(), parent, plane )
54 {
55      init();
56 }
```

7.44.1.2 PieDiagram::∼PieDiagram () `[virtual]`

Definition at line 58 of file KDChartPieDiagram.cpp.

```
59 {
60 }
```

7.44.2 Member Function Documentation

7.44.2.1 bool AbstractDiagram::allowOverlappingDataValueTexts () const `[inherited]`

Returns:

Whether data value labels are allowed to overlap.

Definition at line 446 of file KDChartAbstractDiagram.cpp.

References d.

```
450 {
```

7.44.2.2 bool AbstractDiagram::antiAliasing () const [inherited]

Returns:

Whether anti-aliasing is to be used for rendering this diagram.

Definition at line 457 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::paint().

```
461 {
```

7.44.2.3 AttributesModel ∗ AbstractDiagram::attributesModel () const [virtual, inherited]

Returns the AttributesModel, that is used by this diagram.

By default each diagram owns its own AttributesModel, which should never be deleted. Only if a user-supplied AttributesModel has been set does the pointer returned here not belong to the diagram.

Returns:

The AttributesModel associated with the diagram.

See also:

setAttributesModel

Definition at line 286 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::RingDiagram::paint(), KDChart::PolarDiagram::paint(), and KDChart::BarDiagram::setBarAttributes().

```
287 {
288     return d->attributesModel;
289 }
```

7.44.2.4 QModelIndex AbstractDiagram::attributesModelRootIndex () const [protected, inherited]

returns a QModelIndex pointing into the AttributesModel that corresponds to the root index of the diagram.

Definition at line 310 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::calculateDataBoundaries(), KDChart::BarDiagram::calculateDataBoundaries(), KDChart::LineDiagram::numberOfAbscissaSegments(), KDChart::BarDiagram::numberOfAbscissaSegments(), KDChart::LineDiagram::numberOfOrdinateSegments(), KDChart::BarDiagram::numberOfOrdinateSegments(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and KDChart::AbstractDiagram::valueForCell().

```
316 {
```

7.44.2.5 QBrush AbstractDiagram::brush (const QModelIndex & *index*) const `[inherited]`

Retrieve the brush to be used, for painting the datapoint at the given index in the model.

Parameters:

index The index of the datapoint in the model.

Returns:

The brush to use for painting.

Definition at line 816 of file KDChartAbstractDiagram.cpp.

```
822                                :
QRect AbstractDiagram::visualRect(const QModelIndex &) const
```

7.44.2.6 QBrush AbstractDiagram::brush (int *dataset*) const `[inherited]`

Retrieve the brush to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the brush for.

Returns:

The brush to use for painting.

Definition at line 808 of file KDChartAbstractDiagram.cpp.

```
815 {
```

7.44.2.7 QBrush AbstractDiagram::brush () const `[inherited]`

Retrieve the brush to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The brush to use for painting.

Definition at line 802 of file KDChartAbstractDiagram.cpp.

Referenced by paint(), KDChart::LineDiagram::paint(), and KDChart::AbstractDiagram::paintMarker().

```
807 {
```

7.44.2.8 const QPair< QPointF, QPointF > PieDiagram::calculateDataBoundaries () const `[protected, virtual]`

[reimplemented]

Implements KDChart::AbstractDiagram.

Definition at line 71 of file KDChartPieDiagram.cpp.

References KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractPolarDiagram::columnCount(), KDChart::PieAttributes::explode(), KDChart::PieAttributes::explodeFactor(), and KDChart::AbstractPieDiagram::pieAttributes().

```
72 {
73     if ( !checkInvariants( true ) ) return QPair<QPointF, QPointF>( QPointF( 0, 0 ), QPointF( 0, 0 ) );
74
75     const PieAttributes attrs( pieAttributes( model()->index( 0, 0, rootIndex() ) ) );
76
77     QPointF bottomLeft ( QPointF( 0, 0 ) );
78     QPointF topRight;
79     // If we explode, we need extra space for the pie slice that has
80     // the largest explosion distance.
81     if ( attrs.explode() ) {
82         const int colCount = columnCount();
83         qreal maxExplode = 0.0;
84         for( int j = 0; j < colCount; ++j ){
85             const PieAttributes columnAttrs( pieAttributes( model()->index( 0, j, rootIndex() ) ) );
86             maxExplode = qMax( maxExplode, columnAttrs.explodeFactor() );
87         }
88         topRight = QPointF( 1.0+maxExplode, 1.0+maxExplode );
89     }else{
90         topRight = QPointF( 1.0, 1.0 );
91     }
92     return QPair<QPointF, QPointF> ( bottomLeft,  topRight );
93 }
```

7.44.2.9 bool AbstractDiagram::checkInvariants (bool *justReturnTheStatus* = false) const `[protected, virtual, inherited]`

Definition at line 930 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane().

Referenced by KDChart::RingDiagram::calculateDataBoundaries(), KDChart::PolarDiagram::calculateDataBoundaries(), calculateDataBoundaries(), KDChart::LineDiagram::calculateDataBoundaries(), KDChart::BarDiagram::calculateDataBoundaries(), KDChart::RingDiagram::paint(), KDChart::PolarDiagram::paint(), paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and KDChart::AbstractDiagram::paintMarker().

```
930                                {
931         Q_ASSERT_X ( model(), "AbstractDiagram::checkInvariants()",
932                     "There is no usable model set, for the diagram." );
933
934         Q_ASSERT_X ( coordinatePlane(), "AbstractDiagram::checkInvariants()",
935                     "There is no usable coordinate plane set, for the diagram." );
936     }
937     return model() && coordinatePlane();
938 }
939
940 int AbstractDiagram::datasetDimension( ) const
```

7.44.2.10 PieDiagram ∗ PieDiagram::clone () const `[virtual]`

Definition at line 66 of file KDChartPieDiagram.cpp.

References d, and PieDiagram().

```
67 {
68     return new PieDiagram( new Private( *d ) );
69 }
```

7.44.2.11 int AbstractPolarDiagram::columnCount () const `[inherited]`

Definition at line 60 of file KDChartAbstractPolarDiagram.cpp.

References KDChart::AbstractPolarDiagram::numberOfValuesPerDataset().

Referenced by calculateDataBoundaries(), paint(), and valueTotals().

```
61 {
62     return static_cast<int>( numberOfValuesPerDataset() );
63 }
```

7.44.2.12 QModelIndex AbstractDiagram::columnToIndex (int *column*) const `[protected, inherited]`

Definition at line 317 of file KDChartAbstractDiagram.cpp.

```
323 {
```

7.44.2.13 bool AbstractDiagram::compare (const AbstractDiagram ∗ *other*) const `[inherited]`

Returns true if both diagrams have the same settings.

Definition at line 135 of file KDChartAbstractDiagram.cpp.

```
136 {
137     if( other == this ) return true;
138     if( ! other ){
139         //qDebug() << "AbstractDiagram::compare() cannot compare to Null pointer";
140         return false;
141     }
142     /*
143     qDebug() << "\n             AbstractDiagram::compare() QAbstractScrollArea:";
144             // compare QAbstractScrollArea properties
145     qDebug() <<
146             ((horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
147             (verticalScrollBarPolicy()    == other->verticalScrollBarPolicy()));
148     qDebug() << "AbstractDiagram::compare() QFrame:";
149             // compare QFrame properties
150     qDebug() <<
151             ((frameShadow() == other->frameShadow()) &&
152             (frameShape()   == other->frameShape()) &&
153             (frameWidth()   == other->frameWidth()) &&
154             (lineWidth()    == other->lineWidth()) &&
155             (midLineWidth() == other->midLineWidth()));
```

```
    qDebug() << "AbstractDiagram::compare() QAbstractItemView:";
            // compare QAbstractItemView properties
    qDebug() <<
            ((alternatingRowColors() == other->alternatingRowColors()) &&
            (hasAutoScroll()         == other->hasAutoScroll()) &&
#if QT_VERSION > 0x040199
            (dragDropMode()          == other->dragDropMode()) &&
            (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
            (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
            (verticalScrollMode()    == other->verticalScrollMode()) &&
#endif
            (dragEnabled()           == other->dragEnabled()) &&
            (editTriggers()          == other->editTriggers()) &&
            (iconSize()              == other->iconSize()) &&
            (selectionBehavior()     == other->selectionBehavior()) &&
            (selectionMode()         == other->selectionMode()) &&
            (showDropIndicator()     == other->showDropIndicator()) &&
            (tabKeyNavigation()      == other->tabKeyNavigation()) &&
            (textElideMode()         == other->textElideMode()));
    qDebug() << "AbstractDiagram::compare() AttributesModel: ";
            // compare all of the properties stored in the attributes model
    qDebug() << attributesModel()->compare( other->attributesModel() );
    qDebug() << "AbstractDiagram::compare() own:";
            // compare own properties
    qDebug() <<
            ((rootIndex().column()            == other->rootIndex().column()) &&
            (rootIndex().row()                == other->rootIndex().row()) &&
            (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
            (antiAliasing()                   == other->antiAliasing()) &&
            (percentMode()                    == other->percentMode()) &&
            (datasetDimension()               == other->datasetDimension()));
    */
    return  // compare QAbstractScrollArea properties
            (horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
            (verticalScrollBarPolicy()   == other->verticalScrollBarPolicy()) &&
            // compare QFrame properties
            (frameShadow()  == other->frameShadow()) &&
            (frameShape()   == other->frameShape()) &&
            (frameWidth()   == other->frameWidth()) &&
            (lineWidth()    == other->lineWidth()) &&
            (midLineWidth() == other->midLineWidth()) &&
            // compare QAbstractItemView properties
            (alternatingRowColors()  == other->alternatingRowColors()) &&
            (hasAutoScroll()         == other->hasAutoScroll()) &&
#if QT_VERSION > 0x040199
            (dragDropMode()          == other->dragDropMode()) &&
            (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
            (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
            (verticalScrollMode()    == other->verticalScrollMode()) &&
#endif
            (dragEnabled()           == other->dragEnabled()) &&
            (editTriggers()          == other->editTriggers()) &&
            (iconSize()              == other->iconSize()) &&
            (selectionBehavior()     == other->selectionBehavior()) &&
            (selectionMode()         == other->selectionMode()) &&
            (showDropIndicator()     == other->showDropIndicator()) &&
            (tabKeyNavigation()      == other->tabKeyNavigation()) &&
            (textElideMode()         == other->textElideMode()) &&
            // compare all of the properties stored in the attributes model
            attributesModel()->compare( other->attributesModel() ) &&
            // compare own properties
            (rootIndex().column()             == other->rootIndex().column()) &&
            (rootIndex().row()                == other->rootIndex().row()) &&
            (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
            (antiAliasing()                   == other->antiAliasing()) &&
            (percentMode()                    == other->percentMode()) &&
            (datasetDimension()               == other->datasetDimension());
```

```
}
```

7.44.2.14 AbstractCoordinatePlane * AbstractDiagram::coordinatePlane () const `[inherited]`

The coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. By default this is a CartesianCoordinatePlane.

Returns:

The coordinate plane associated with the diagram.

Definition at line 226 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractCartesianDiagram::layoutPlanes(), KDChart::PolarDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), KDChart::AbstractPolarDiagram::polarCoordinatePlane(), and KDChart::AbstractCartesianDiagram::setCoordinatePlane().

```
{
    return d->plane;
}
```

7.44.2.15 const QPair< QPointF, QPointF > AbstractDiagram::dataBoundaries () const `[inherited]`

Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

This method returns a chached result of calculations done by calculateDataBoundaries. Classes derived from AbstractDiagram must implement the calculateDataBoundaries function, to specify their own way of calculating the data boundaries. If derived classes want to force recalculation of the data boundaries, they can call setDataBoundariesDirty()

Returned value is in diagram coordinates.

Definition at line 231 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::calculateDataBoundaries(), and d.

Referenced by KDChart::CartesianCoordinatePlane::getRawDataBoundingRectFromDiagrams(), KDChart::PolarCoordinatePlane::layoutDiagrams(), KDChart::LineDiagram::paint(), and KDChart::BarDiagram::paint().

```
{
    if( d->databoundariesDirty ){
        d->databoundaries = calculateDataBoundaries ();
        d->databoundariesDirty = false;
    }
    return d->databoundaries;
}
```

7.44.2.16 void AbstractDiagram::dataChanged (const QModelIndex & *topLeft*, const QModelIndex & *bottomRight*) `[virtual, inherited]`

[reimplemented]

Definition at line 338 of file KDChartAbstractDiagram.cpp.

References d.

```
338 {
339   // We are still too dumb to do intelligent updates...
340   d->databoundariesDirty = true;
341   scheduleDelayedItemsLayout();
342 }
343
344
```

7.44.2.17 void KDChart::AbstractDiagram::dataHidden () `[protected, inherited]`

This signal is emitted, when the hidden status of at least one data cell was (un)set.

7.44.2.18 QList< QBrush > AbstractDiagram::datasetBrushes () const `[inherited]`

The set of dataset brushes currently used, for use in legends, etc.

Note:

Cell-level override brushes, if set, take precedence over the dataset values, so you might need to check these too, in order to find the brush, that is used for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 894 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), KDChart::Legend::datasetCount(), and KDChart::Legend::setBrushesFromDiagram().

```
896                                                                                  {
897         QBrush brush = qVariantValue<QBrush>( attributesModel()->headerData( i, Qt::Vertical, DatasetB
898         ret << brush;
899     }
900
901     return ret;
902 }
903
904 QList<QPen> AbstractDiagram::datasetPens() const
```

7.44.2.19 int AbstractDiagram::datasetDimension () const `[inherited]`

The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

For each dimension it will expect one column of values in the model. If the dimensionality is 1, automatic values will be used for the abscissa.

For example a diagram with the default dimension of 1, will have one column per datapoint (the y values) and will use automatic values for the x axis (1, 2, 3, ... n). If the dimension is 2, the diagram will use the first, (and the third, fifth, etc) columns as X values, and the second, (and the fourth, sixth, etc) column as Y values.

Returns:

The dataset dimension of the diagram.

Definition at line 942 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::calculateDataBoundaries(), KDChart::LineDiagram::getCellValues(), KDChart::CartesianCoordinatePlane::getDataDimensionsList(), KDChart::LineDiagram::paint(), and KDChart::LineDiagram::setType().

```
946 {
```

7.44.2.20 QStringList AbstractDiagram::datasetLabels () const `[inherited]`

The set of dataset labels currently displayed, for use in legends, etc.

Returns:

The set of dataset labels currently displayed.

Definition at line 882 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), and KDChart::Legend::datasetCount().

```
883                                                  : " << attributesModel()->columnCount(attributesModel
884     const int columnCount = attributesModel()->columnCount(attributesModelRootIndex());
885     for( int i = datasetDimension()-1; i < columnCount; i += datasetDimension() ){
886         //qDebug() << "dataset label: " << attributesModel()->headerData( i, Qt::Horizontal, Qt::Displ
887         ret << attributesModel()->headerData( i, Qt::Horizontal, Qt::DisplayRole ).toString();
888     }
889     return ret;
890 }
891
892 QList<QBrush> AbstractDiagram::datasetBrushes() const
```

7.44.2.21 QList< MarkerAttributes > AbstractDiagram::datasetMarkers () const `[inherited]`

The set of dataset markers currently used, for use in legends, etc.

Note:

Cell-level override markers, if set, take precedence over the dataset values, so you might need to check these too, in order to find the marker, that is shown for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 917 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
                                                                                  {
        DataValueAttributes a =
            qVariantValue<DataValueAttributes>( attributesModel()->headerData( i, Qt::Vertical, DataVa
        const MarkerAttributes &ma = a.markerAttributes();
        ret << ma;
    }
    return ret;
}

bool AbstractDiagram::checkInvariants( bool justReturnTheStatus ) const
```

7.44.2.22 QList< QPen > AbstractDiagram::datasetPens () const [inherited]

The set of dataset pens currently used, for use in legends, etc.

Note:

Cell-level override pens, if set, take precedence over the dataset values, so you might need to check these too, in order to find the pens, that is used for a single cell.

Returns:

The current set of dataset pens.

Definition at line 906 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
                                                                                  {
        QPen pen = qVariantValue<QPen>( attributesModel()->headerData( i, Qt::Vertical, DatasetPenRole
        ret << pen;
    }
    return ret;
}

QList<MarkerAttributes> AbstractDiagram::datasetMarkers() const
```

7.44.2.23 DataValueAttributes AbstractDiagram::dataValueAttributes (const QModelIndex & *index*) const [inherited]

Retrieve the DataValueAttributes for the given index.

This will fall back automatically to what was set at dataset or model level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the attributes for.

Returns:

The DataValueAttributes for the given index.

Definition at line 427 of file KDChartAbstractDiagram.cpp.

```
{
```

7.44.2.24 DataValueAttributes AbstractDiagram::dataValueAttributes (int *column*) const `[inherited]`

Retrieve the DataValueAttributes for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the attributes for.

Returns:

The DataValueAttributes for the given dataset.

Definition at line 420 of file KDChartAbstractDiagram.cpp.

```
426 {
```

7.44.2.25 DataValueAttributes AbstractDiagram::dataValueAttributes () const `[inherited]`

Retrieve the DataValueAttributes speficied globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The global DataValueAttributes.

Definition at line 414 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::AbstractDiagram::paintDataValueText(), and KDChart::AbstractDiagram::paintMarker().

```
419 {
```

7.44.2.26 void AbstractDiagram::doItemsLayout () `[virtual, inherited]`

[reimplemented]

Definition at line 329 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::AbstractDiagram::update().

```
329                     {
330         d->plane->layoutDiagrams();
331         update();
332     }
333     QAbstractItemView::doItemsLayout();
334 }
335
336 void AbstractDiagram::dataChanged( const QModelIndex &topLeft,
```

7.44.2.27 qreal AbstractPieDiagram::granularity () const `[inherited]`

Returns:

the granularity.

Definition at line 69 of file KDChartAbstractPieDiagram.cpp.

References d.

Referenced by paint().

```
{
    return (d->granularity < 0.05 || d->granularity > 36.0)
            ? 1.0
    : d->granularity;
}
```

7.44.2.28 int AbstractDiagram::horizontalOffset () const `[virtual, inherited]`

[reimplemented]

Definition at line 839 of file KDChartAbstractDiagram.cpp.

```
{ return 0; }
```

7.44.2.29 QModelIndex AbstractDiagram::indexAt (const QPoint & *point*) const `[virtual, inherited]`

[reimplemented]

Definition at line 833 of file KDChartAbstractDiagram.cpp.

```
{ return QModelIndex(); }
```

7.44.2.30 bool AbstractDiagram::isHidden (const QModelIndex & *index*) const `[inherited]`

Retrieve the hidden status for the given index.

This will fall back automatically to what was set at dataset or diagram level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the hidden status for.

Returns:

The hidden status for the given index.

Definition at line 386 of file KDChartAbstractDiagram.cpp.

7.44.2.31 bool AbstractDiagram::isHidden (int *column*) const `[inherited]`

Retrieve the hidden status for the given dataset.

This will fall back automatically to what was set at diagram level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the hidden status for.

Returns:

The hidden status for the given dataset.

Definition at line 379 of file KDChartAbstractDiagram.cpp.

```
385 {
```

7.44.2.32 bool AbstractDiagram::isHidden () const `[inherited]`

Retrieve the hidden status speficied globally.

This will fall back automatically to the default settings (= not hidden), if there are no specific settings.

Returns:

The global hidden status.

Definition at line 373 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), KDChart::LineDiagram::paint(), and KDChart::LineDiagram::valueForCellTesting().

```
378 {
```

7.44.2.33 bool AbstractDiagram::isIndexHidden (const QModelIndex & *index*) const `[virtual, inherited]`

[reimplemented]

Definition at line 845 of file KDChartAbstractDiagram.cpp.

```
847 {}
```

7.44.2.34 QStringList AbstractDiagram::itemRowLabels () const `[inherited]`

The set of item row labels currently displayed, for use in Abscissa axes, etc.

Returns:

The set of item row labels currently displayed.

Definition at line 870 of file KDChartAbstractDiagram.cpp.

```
871                                                  : " << attributesModel()->rowCount(attributesModelRoo
872     const int rowCount = attributesModel()->rowCount(attributesModelRootIndex());
873     for( int i = 0; i < rowCount; ++i ){
874         //qDebug() << "item row label: " << attributesModel()->headerData( i, Qt::Vertical, Qt::Displa
875         ret << attributesModel()->headerData( i, Qt::Vertical, Qt::DisplayRole ).toString();
876     }
877     return ret;
878 }
879
880 QStringList AbstractDiagram::datasetLabels() const
```

7.44.2.35 void KDChart::AbstractDiagram::modelsChanged () `[protected, inherited]`

This signal is emitted, when either the model or the AttributesModel is replaced.

Referenced by KDChart::AbstractDiagram::setAttributesModel(), and KDChart::AbstractDiagram::setModel().

7.44.2.36 QModelIndex AbstractDiagram::moveCursor (CursorAction *cursorAction*, Qt::KeyboardModifiers *modifiers*) `[virtual, inherited]`

[reimplemented]

Definition at line 836 of file KDChartAbstractDiagram.cpp.

```
838 { return 0; }
```

7.44.2.37 double PieDiagram::numberOfGridRings () const `[virtual]`

[reimplemented]

Implements KDChart::AbstractPolarDiagram.

Definition at line 1107 of file KDChartPieDiagram.cpp.

```
1108 {
1109     return 1;
1110 }
```

7.44.2.38 double PieDiagram::numberOfValuesPerDataset () const `[virtual]`

[reimplemented]

Implements KDChart::AbstractPolarDiagram.

Definition at line 1101 of file KDChartPieDiagram.cpp.

```
1102 {
1103     return model() ? model()->columnCount( rootIndex() ) : 0.0;
1104 }
```

7.44.2.39 void PieDiagram::paint (PaintContext ∗ *paintContext*) `[protected, virtual]`

[reimplemented]

Implements KDChart::AbstractDiagram.

Definition at line 169 of file KDChartPieDiagram.cpp.

References KDChart::AbstractDiagram::brush(), buildReferenceRect(), KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractPolarDiagram::columnCount(), d, KDChart::AbstractThreeDAttributes::depth(), KDChart::PieAttributes::explodeFactor(), KDChart::AbstractPieDiagram::granularity(), KDChart::AbstractThreeDAttributes::isEnabled(), KDChart::PaintContext::painter(), KDChart::AbstractDiagram::pen(), KDChart::AbstractPieDiagram::pieAttributes(), KDChart::AbstractPolarDiagram::polarCoordinatePlane(), KDChart::PaintContext::rectangle(), KDChart::PolarCoordinatePlane::startPosition(), KDChart::AbstractPieDiagram::threeDPieAttributes(), and valueTotals().

Referenced by paintEvent().

```
{
    // note: Not having any data model assigned is no bug
    //       but we can not draw a diagram then either.
    if ( !checkInvariants(true) )
        return;

    const PieAttributes attrs( pieAttributes() );
    const ThreeDPieAttributes threeDAttrs( threeDPieAttributes( model()->index( 0, 0, rootIndex() ) )

    const int colCount = columnCount();

    QRectF contentsRect( buildReferenceRect( polarCoordinatePlane() ) );
    contentsRect = ctx->rectangle();
//    contentsRect = geometry();
//qDebug() << contentsRect;
    if( contentsRect.isEmpty() )
        return;

    DataValueTextInfoList list;
    const qreal sum = valueTotals();

    if( sum == 0.0 ) //nothing to draw
        return;

    d->startAngles.resize( colCount );
    d->angleLens.resize( colCount );

    // compute position
    d->size = qMin( contentsRect.width(), contentsRect.height() ); // initial size

    // if the pies explode, we need to give them additional space =>
    // make the basic size smaller
    qreal maxExplode = 0.0;
    for( int j = 0; j < colCount; ++j ){
        const PieAttributes columnAttrs( pieAttributes( model()->index( 0, j, rootIndex() ) ) );
        maxExplode = qMax( maxExplode, columnAttrs.explodeFactor() );
    }
    d->size /= ( 1.0 + 2.0 * maxExplode );


    qreal sizeFor3DEffect = 0.0;
    if ( ! threeDAttrs.isEnabled() ) {

        qreal x = ( contentsRect.width() == d->size ) ? 0.0 : ( ( contentsRect.width() - d->size ) / 2
        qreal y = ( contentsRect.height() == d->size ) ? 0.0 : ( ( contentsRect.height() - d->size ) /
        d->position = QRectF( x, y, d->size, d->size );
        d->position.translate( contentsRect.left(), contentsRect.top() );
    } else {
        // threeD: width is the maximum possible width; height is 1/2 of that
        qreal x = ( contentsRect.width() == d->size ) ? 0.0 : ( ( contentsRect.width() - d->size ) / 2
        qreal height = d->size;
        // make sure that the height plus the threeDheight is not more than the
        // available size
        if ( threeDAttrs.depth() >= 0.0 ) {
            // positive pie height: absolute value
            sizeFor3DEffect = threeDAttrs.depth();
            height = d->size - sizeFor3DEffect;
        } else {
            // negative pie height: relative value
            sizeFor3DEffect = - threeDAttrs.depth() / 100.0 * height;
            height = d->size - sizeFor3DEffect;
        }
        qreal y = ( contentsRect.height() == height ) ? 0.0 : ( ( contentsRect.height() - height - siz

        d->position = QRectF( contentsRect.left() + x, contentsRect.top() + y,
                d->size, height );
        //  d->position.moveBy( contentsRect.left(), contentsRect.top() );
    }

    const PolarCoordinatePlane * plane = polarCoordinatePlane();
    const qreal sectorsPerValue = 360.0 / sum;
    qreal currentValue = plane ? plane->startPosition() : 0.0;

    bool atLeastOneValue = false; // guard against completely empty tables
    QVariant vValY;
    for ( int iColumn = 0; iColumn < colCount; ++iColumn ) {
        // is there anything at all at this column?
        bool bOK;
        const double cellValue = qAbs( model()->data( model()->index( 0, iColumn, rootIndex() ) )
            .toDouble( &bOK ) );

        if( bOK ){
            d->startAngles[ iColumn ] = currentValue;
            d->angleLens[ iColumn ] = cellValue * sectorsPerValue;
            atLeastOneValue = true;
        } else { // mark as non-existent
            d->angleLens[ iColumn ] = 0.0;
            if ( iColumn > 0.0 )
                d->startAngles[ iColumn ] = d->startAngles[ iColumn - 1 ];
            else
                d->startAngles[ iColumn ] = currentValue;
        }
        //qDebug() << "d->startAngles["<<iColumn<<"] == " << d->startAngles[ iColumn ]
        //         << " +  d->angleLens["<<iColumn<<"]" << d->angleLens[ iColumn ]
        //         << " = " << d->startAngles[ iColumn ]+d->angleLens[ iColumn ];

        currentValue = d->startAngles[ iColumn ] + d->angleLens[ iColumn ];
    }

    // If there was no value at all, bail out, to avoid endless loops
    // later on (e.g. in findPieAt()).
    if( ! atLeastOneValue )
        return;


    // Find the backmost pie which is at +90Âř and needs to be drawn
    // first
    int backmostpie = findPieAt( 90, colCount );
    // Find the frontmost pie (at -90Âř/+270Âř) that should be drawn last
    int frontmostpie = findPieAt( 270, colCount );
    // the right- and the leftmost (only needed in some special cases...)
    int rightmostpie = findPieAt( 0, colCount );
    int leftmostpie = findPieAt( 180, colCount );


    int currentLeftPie = backmostpie;
    int currentRightPie = backmostpie;

    drawOnePie( ctx->painter(), 0, backmostpie, granularity(), sizeFor3DEffect );

    if( backmostpie == frontmostpie )
    {
        if( backmostpie == leftmostpie )
            currentLeftPie = findLeftPie( currentLeftPie, colCount );
        if( backmostpie == rightmostpie )
            currentRightPie = findRightPie( currentRightPie, colCount );
    }
    while( currentLeftPie != frontmostpie )
    {
        if( currentLeftPie != backmostpie )
            drawOnePie( ctx->painter(), 0, currentLeftPie, granularity(), sizeFor3DEffect );
        currentLeftPie = findLeftPie( currentLeftPie, colCount );
    }
    while( currentRightPie != frontmostpie )
    {
        if( currentRightPie != backmostpie )
            drawOnePie( ctx->painter(), 0, currentRightPie, granularity(), sizeFor3DEffect );
        currentRightPie = findRightPie( currentRightPie, colCount );
    }

    // if the backmost pie is not the frontmost pie, we draw the frontmost at last
    if( backmostpie != frontmostpie || ! threeDPieAttributes().isEnabled() )
    {
        drawOnePie( ctx->painter(), 0, frontmostpie, granularity(), sizeFor3DEffect );
    // else, this gets a bit mor complicated...
    } else if( threeDPieAttributes().isEnabled() ) {
        drawPieSurface( ctx->painter(), 0, frontmostpie, granularity() );
        const QModelIndex index = model()->index( 0, frontmostpie, rootIndex() );
        QPen pen = this->pen( index );
        ctx->painter()->setRenderHint ( QPainter::Antialiasing );
        ctx->painter()->setBrush( brush( index ) );
        if ( threeDAttrs.isEnabled() )
            pen.setColor( QColor( 0, 0, 0 ) );
        ctx->painter()->setPen( pen );

        qreal startAngle = d->startAngles[ frontmostpie ];
        if( startAngle > 360 )
            startAngle -= 360;

        qreal endAngle = startAngle + d->angleLens[ frontmostpie ];
        startAngle = qMax( startAngle, 180.0 );

        drawArcEffectSegment( ctx->painter(), piePosition( 0, frontmostpie),
                sizeFor3DEffect, startAngle, endAngle, granularity() );
    }
}
```

7.44.2.40 void AbstractDiagram::paintDataValueText (QPainter ∗ *painter*, const QModelIndex & *index*, const QPointF & *pos*, double *value*) `[inherited]`

Definition at line 474 of file KDChartAbstractDiagram.cpp.

References KDChart::RelativePosition::alignment(), KDChart::TextAttributes::calculatedFont(), d, KDChart::DataValueAttributes::dataLabel(), KDChart::AbstractDiagram::dataValueAttributes(), KDChart::DataValueAttributes::decimalDigits(), KDChart::TextAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::TextAttributes::pen(), KDChart::DataValueAttributes::position(), KDChart::DataValueAttributes::prefix(), KDChart::TextAttributes::rotation(), KDChart::DataValue-

Attributes::showRepetitiveDataLabels(), KDChart::DataValueAttributes::suffix(), and KDChart::DataValueAttributes::textAttributes().

Referenced by KDChart::RingDiagram::paint(), and KDChart::PolarDiagram::paint().

```
{
    // paint one data series
    const DataValueAttributes a( dataValueAttributes(index) );
    if ( !a.isVisible() ) return;

    // handle decimal digits
    int decimalDigits = a.decimalDigits();
    int decimalPos = QString::number(  value ).indexOf( QLatin1Char( ’.’ ) );
    QString roundedValue;
    if ( a.dataLabel().isNull() ) {
        if ( decimalPos > 0 && value != 0 )
            roundedValue =  roundValues ( value, decimalPos, decimalDigits );
        else
            roundedValue = QString::number(  value );
    } else
        roundedValue = a.dataLabel();
        // handle prefix and suffix
    if ( !a.prefix().isNull() )
        roundedValue.prepend( a.prefix() );

    if ( !a.suffix().isNull() )
        roundedValue.append( a.suffix() );

    const TextAttributes ta( a.textAttributes() );
    // FIXME draw the non-text bits, background, etc
    if ( ta.isVisible() ) {

        QPointF pt( pos );
        /* for debugging:
        PainterSaver painterSaver( painter );
        painter->setPen( Qt::black );
        painter->drawLine( pos - QPointF( 1,1), pos + QPointF( 1,1) );
        painter->drawLine( pos - QPointF(-1,1), pos + QPointF(-1,1) );
        */

        // adjust the text start point position, if alignment is not Bottom/Left
        const RelativePosition relPos( a.position( value >= 0.0 ) );
        const Qt::Alignment alignBottomLeft = Qt::AlignBottom | Qt::AlignLeft;
        const QFont calculatedFont( ta.calculatedFont( d->plane, KDChartEnums::MeasureOrientationMinim
        //qDebug() << "calculatedFont’s point size:" << calculatedFont.pointSizeF();
        if( (relPos.alignment() & alignBottomLeft) != alignBottomLeft ){
            const QRectF boundRect(
                    d->cachedFontMetrics( calculatedFont, this )->boundingRect( roundedValue ) );
            if( relPos.alignment() & Qt::AlignRight )
                pt.rx() -= boundRect.width();
            else if( relPos.alignment() & Qt::AlignHCenter )
                pt.rx() -= 0.5 * boundRect.width();

            if( relPos.alignment() & Qt::AlignTop )
                pt.ry() += boundRect.height();
            else if( relPos.alignment() & Qt::AlignVCenter )
                pt.ry() += 0.5 * boundRect.height();
        }

        // FIXME draw the non-text bits, background, etc

        if ( a.showRepetitiveDataLabels() ||
             pos.x() <= d->lastX ||
             d->lastRoundedValue != roundedValue ) {
            d->lastRoundedValue = roundedValue;
            d->lastX = pos.x();

538             PainterSaver painterSaver( painter );
539             painter->setPen( ta.pen() );
540             painter->setFont( calculatedFont );
541             painter->translate( pt );
542             painter->rotate( ta.rotation() );
543             painter->drawText( QPointF(0, 0), roundedValue );
544         }
545     }
546 }
547 
548 
```

7.44.2.41 void AbstractDiagram::paintDataValueTexts (QPainter ∗ *painter*) `[protected, virtual, inherited]`

Definition at line 576 of file KDChartAbstractDiagram.cpp.

```
579                                                                                {
580        for ( int j=0; j< rowCount; ++j ) {
581            const QModelIndex index = model()->index( j, i, rootIndex() );
582            double value = model()->data( index ).toDouble();
583            const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
584            paintDataValueText( painter, index, pos, value );
585        }
586     }
587 }
588 
589 
```

7.44.2.42 void PieDiagram::paintEvent (QPaintEvent ∗) `[protected]`

Definition at line 96 of file KDChartPieDiagram.cpp.

References paint(), KDChart::PaintContext::setPainter(), and KDChart::PaintContext::setRectangle().

```
97 {
98     QPainter painter ( viewport() );
99     PaintContext ctx;
100     ctx.setPainter ( &painter );
101     ctx.setRectangle( QRectF ( 0, 0, width(), height() ) );
102     paint ( &ctx );
103 }
```

7.44.2.43 void AbstractDiagram::paintMarker (QPainter ∗ *painter*, const QModelIndex & *index*, const QPointF & *pos*) `[inherited]`

Definition at line 592 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::brush(), KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractDiagram::dataValueAttributes(), KDChart::MarkerAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::DataValueAttributes::markerAttributes(), KDChart::MarkerAttributes::markerColor(), KDChart::MarkerAttributes::markerSize(), KDChart::AbstractDiagram::paintMarker(), and KDChart::MarkerAttributes::pen().

```
593 {
594 
```

```
    if ( !checkInvariants() ) return;
    DataValueAttributes a = dataValueAttributes(index);
    if ( !a.isVisible() ) return;
    const MarkerAttributes &ma = a.markerAttributes();
    if ( !ma.isVisible() ) return;

    PainterSaver painterSaver( painter );
    QSizeF maSize( ma.markerSize() );
    QBrush indexBrush( brush( index ) );
    QPen indexPen( ma.pen() );
    if ( ma.markerColor().isValid() )
        indexBrush.setColor( ma.markerColor() );

    paintMarker( painter, ma, indexBrush, indexPen, pos, maSize );
}
```

7.44.2.44 void AbstractDiagram::paintMarker (QPainter ∗ *painter*, const MarkerAttributes & *markerAttributes*, const QBrush & *brush*, const QPen &, const QPointF & *point*, const QSizeF & *size*) [virtual, inherited]

Definition at line 614 of file KDChartAbstractDiagram.cpp.

References KDChart::MarkerAttributes::markerStyle().

Referenced by KDChart::MarkerLayoutItem::paintIntoRect(), and KDChart::AbstractDiagram::paintMarker().

```
{

    const QPen oldPen( painter->pen() );
    // Pen is used to paint 4Pixels - 1 Pixel - Ring and FastCross types.
    // make sure to use the brush color - see above in those cases.
    const bool isFourPixels = (markerAttributes.markerStyle() == MarkerAttributes::Marker4Pixels);
    if( isFourPixels || (markerAttributes.markerStyle() == MarkerAttributes::Marker1Pixel) ){
        // for high-performance point charts with tiny point markers:
        painter->setPen( QPen( brush.color().light() ) );
        if( isFourPixels ){
            const qreal x = pos.x();
            const qreal y = pos.y();
            painter->drawLine( QPointF(x-1.0,y-1.0),
                               QPointF(x+1.0,y-1.0) );
            painter->drawLine( QPointF(x-1.0,y),
                               QPointF(x+1.0,y) );
            painter->drawLine( QPointF(x-1.0,y+1.0),
                               QPointF(x+1.0,y+1.0) );
        }
        painter->drawPoint( pos );
    }else{
        PainterSaver painterSaver( painter );
        // we only a solid line surrounding the markers
        QPen painterPen( pen );
        painterPen.setStyle( Qt::SolidLine );
        painter->setPen( painterPen );
        painter->setBrush( brush );
        painter->setRenderHint ( QPainter::Antialiasing );
        painter->translate( pos );
        switch ( markerAttributes.markerStyle() ) {
            case MarkerAttributes::MarkerCircle:
                painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                            maSize.height(), maSize.width()) );
                break;
            case MarkerAttributes::MarkerSquare:
                {
                    QRectF rect( 0 - maSize.width()/2, 0 - maSize.height()/2,
                                maSize.width(), maSize.height() );
                    painter->drawRect( rect );
                    painter->fillRect( rect, brush.color() );
                    break;
                }
            case MarkerAttributes::MarkerDiamond:
                {
                    QVector <QPointF > diamondPoints;
                    QPointF top, left, bottom, right;
                    top    = QPointF( 0, 0 - maSize.height()/2 );
                    left   = QPointF( 0 - maSize.width()/2, 0 );
                    bottom = QPointF( 0, maSize.height()/2 );
                    right  = QPointF( maSize.width()/2, 0 );
                    diamondPoints << top << left << bottom << right;
                    painter->drawPolygon( diamondPoints );
                    break;
                }
            // both handled on top of the method:
            case MarkerAttributes::Marker1Pixel:
            case MarkerAttributes::Marker4Pixels:
                    break;
            case MarkerAttributes::MarkerRing:
                {
                    painter->setPen( QPen( brush.color() ) );
                    painter->setBrush( Qt::NoBrush );
                    painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                                        maSize.height(), maSize.width()) );
                    break;
                }
            case MarkerAttributes::MarkerCross:
                {
                    QRectF rect( maSize.width()*-0.5, maSize.height()*-0.2,
                                 maSize.width(), maSize.height()*0.4 );
                    painter->drawRect( rect );
                    rect.setTopLeft(QPointF( maSize.width()*-0.2, maSize.height()*-0.5 ));
                    rect.setSize(QSizeF( maSize.width()*0.4, maSize.height() ));
                    painter->drawRect( rect );
                    break;
                }
            case MarkerAttributes::MarkerFastCross:
                {
                    QPointF left, right, top, bottom;
                    left  = QPointF( -maSize.width()/2, 0 );
                    right = QPointF( maSize.width()/2, 0 );
                    top   = QPointF( 0, -maSize.height()/2 );
                    bottom= QPointF( 0, maSize.height()/2 );
                    painter->setPen( QPen( brush.color() ) );
                    painter->drawLine( left, right );
                    painter->drawLine(  top, bottom );
                    break;
                }
            default:
                Q_ASSERT_X ( false, "paintMarkers()",
                            "Type item does not match a defined Marker Type." );
        }
    }
    painter->setPen( oldPen );
}

void AbstractDiagram::paintMarkers( QPainter* painter )
```

7.44.2.45 void AbstractDiagram::paintMarkers (QPainter ∗ *painter*) `[protected, virtual, inherited]`

Definition at line 716 of file KDChartAbstractDiagram.cpp.

```
                                                                                {
        for ( int j=0; j< rowCount; ++j ) {
            const QModelIndex index = model()->index( j, i, rootIndex() );
            double value = model()->data( index ).toDouble();
            const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
            paintMarker( painter, index, pos );
        }
    }
}


```

7.44.2.46 QPen AbstractDiagram::pen (const QModelIndex & *index*) const `[inherited]`

Retrieve the pen to be used, for painting the datapoint at the given index in the model.

Parameters:

index The index of the datapoint in the model.

Returns:

The pen to use for painting.

Definition at line 770 of file KDChartAbstractDiagram.cpp.

```
{
```

7.44.2.47 QPen AbstractDiagram::pen (int *dataset*) const `[inherited]`

Retrieve the pen to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the pen for.

Returns:

The pen to use for painting.

Definition at line 762 of file KDChartAbstractDiagram.cpp.

```
{
```

7.44.2.48 QPen AbstractDiagram::pen () const `[inherited]`

Retrieve the pen to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:
The pen to use for painting.

Definition at line 756 of file KDChartAbstractDiagram.cpp.

Referenced by paint(), and KDChart::LineDiagram::paint().

```
761 {
```

7.44.2.49 bool AbstractDiagram::percentMode () const `[inherited]`

Definition at line 468 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::CartesianCoordinatePlane::getDataDimensionsList().

7.44.2.50 PieAttributes AbstractPieDiagram::pieAttributes (const QModelIndex & *index*) const `[inherited]`

Definition at line 121 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::PieAttributesRole.

```
122 {
123     return qVariantValue<PieAttributes>(
124         d->attributesModel->data(
125             d->attributesModel->mapFromSource( index ),
126             PieAttributesRole ) );
127 }
```

7.44.2.51 PieAttributes AbstractPieDiagram::pieAttributes (int *column*) const `[inherited]`

Definition at line 113 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::PieAttributesRole.

```
114 {
115     return qVariantValue<PieAttributes>(
116         d->attributesModel->data(
117             d->attributesModel->mapFromSource( columnToIndex( column ) ).column(),
118             PieAttributesRole ) );
119 }
```

7.44.2.52 PieAttributes AbstractPieDiagram::pieAttributes () const `[inherited]`

Definition at line 104 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::PieAttributesRole.

Referenced by calculateDataBoundaries(), and paint().

```
105 {
106     return qVariantValue<PieAttributes>(
107         d->attributesModel->data( PieAttributesRole ) );
108 }
```

7.44.2.53 const PolarCoordinatePlane ∗ AbstractPolarDiagram::polarCoordinatePlane () const `[inherited]`

Definition at line 55 of file KDChartAbstractPolarDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane().

Referenced by paint().

```
56 {
57     return dynamic_cast<const PolarCoordinatePlane*>( coordinatePlane() );
58 }
```

7.44.2.54 void KDChart::AbstractDiagram::propertiesChanged () `[protected, inherited]`

Emitted upon change of a property of the Diagram.

Referenced by KDChart::LineDiagram::resetLineAttributes(), KDChart::AbstractDiagram::setDataValueAttributes(), KDChart::LineDiagram::setLineAttributes(), KDChart::LineDiagram::setThreeDLineAttributes(), and KDChart::LineDiagram::setType().

7.44.2.55 void PieDiagram::resize (const QSizeF & *area*) `[virtual]`

[reimplemented]

Implements KDChart::AbstractDiagram.

Definition at line 109 of file KDChartPieDiagram.cpp.

```
110 {
111 }
```

7.44.2.56 void PieDiagram::resizeEvent (QResizeEvent ∗) `[protected]`

Definition at line 105 of file KDChartPieDiagram.cpp.

```
106 {
107 }
```

7.44.2.57 void AbstractDiagram::scrollTo (const QModelIndex & *index*, ScrollHint *hint* = EnsureVisible) `[virtual, inherited]`

[reimplemented]

Definition at line 830 of file KDChartAbstractDiagram.cpp.

```
832 { return QModelIndex(); }
```

7.44.2.58 void AbstractDiagram::setAllowOverlappingDataValueTexts (bool *allow*) `[inherited]`

Set whether data value labels are allowed to overlap.

Parameters:

allow True means that overlapping labels are allowed.

Definition at line 440 of file KDChartAbstractDiagram.cpp.

References d.

```
445 {
```

7.44.2.59 void AbstractDiagram::setAntiAliasing (bool *enabled*) `[inherited]`

Set whether anti-aliasing is to be used while rendering this diagram.

Parameters:

enabled True means that AA is enabled.

Definition at line 451 of file KDChartAbstractDiagram.cpp.

References d.

```
456 {
```

7.44.2.60 void AbstractDiagram::setAttributesModel (AttributesModel ∗ *model*) `[virtual, inherited]`

Associate an AttributesModel with this diagram.

Note that the diagram does _not_ take ownership of the AttributesModel. This should thus only be used with AttributesModels that have been explicitely created by the user, and are owned by her. Setting an AttributesModel that is internal to another diagram is an error.

Correct:

```
AttributesModel *am = new AttributesModel( model, 0 );
diagram1->setAttributesModel( am );
diagram2->setAttributesModel( am );
```

Wrong:

```
diagram1->setAttributesModel( diagram2->attributesModel() );
```

Parameters:

model The AttributesModel to use for this diagram.

See also:

AttributesModel, usesExternalAttributesModel

Definition at line 261 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::AbstractDiagram::modelsChanged().

```
{
    if( amodel->sourceModel() != model() ) {
        qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
                 "Trying to set an attributesmodel which works on a different "
                 "model than the diagram.");
        return;
    }
    if( qobject_cast<PrivateAttributesModel*>(amodel) ) {
        qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
                 "Trying to set an attributesmodel that is private to another diagram.");
        return;
    }
    d->setAttributesModel(amodel);
    scheduleDelayedItemsLayout();
    d->databoundariesDirty = true;
    emit modelsChanged();
}
```

7.44.2.61 void AbstractDiagram::setAttributesModelRootIndex (const QModelIndex & *idx*) `[protected, inherited]`

Definition at line 301 of file KDChartAbstractDiagram.cpp.

References d.

7.44.2.62 void AbstractDiagram::setBrush (const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting all datasets in the model.

Parameters:

brush The brush to use.

Definition at line 786 of file KDChartAbstractDiagram.cpp.

```
{
```

7.44.2.63 void AbstractDiagram::setBrush (int *dataset*, const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting the given dataset.

Parameters:

dataset The dataset's column in the model.

pen The brush to use.

Definition at line 793 of file KDChartAbstractDiagram.cpp.

```
{
```

7.44.2.64 void AbstractDiagram::setBrush (const QModelIndex & *index*, const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting the datapoint at the given index.

Parameters:

index The datapoint's index in the model.

brush The brush to use.

Definition at line 778 of file KDChartAbstractDiagram.cpp.

```
785 {
```

7.44.2.65 void AbstractDiagram::setCoordinatePlane (AbstractCoordinatePlane ∗ *plane*) `[virtual, inherited]`

Set the coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. The chart takes ownership.

Returns:

The coordinate plane associated with the diagram.

Reimplemented in KDChart::AbstractCartesianDiagram.

Definition at line 324 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractCoordinatePlane::addDiagram(), KDChart::AbstractCartesianDiagram::setCoordinatePlane(), and KDChart::AbstractCoordinatePlane::takeDiagram().

```
328 {
```

7.44.2.66 void AbstractDiagram::setDataBoundariesDirty () const `[protected, inherited]`

Definition at line 240 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::BarDiagram::setThreeDBarAttributes(), KDChart::LineDiagram::setThreeDLineAttributes(), KDChart::LineDiagram::setType(), and KDChart::BarDiagram::setType().

```
241 {
242     d->databoundariesDirty = true;
243 }
```

7.44.2.67 void AbstractDiagram::setDatasetDimension (int *dimension*) `[inherited]`

Sets the dataset dimension of the diagram.

See also:
datasetDimension.

Parameters:
dimension

Definition at line 947 of file KDChartAbstractDiagram.cpp.

References d.

```
954 {
```

7.44.2.68 void AbstractDiagram::setDataValueAttributes (const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for all datapoints in the model.

Parameters:
a The attributes to set.

Definition at line 434 of file KDChartAbstractDiagram.cpp.

References d.

```
439 {
```

7.44.2.69 void AbstractDiagram::setDataValueAttributes (int *dataset*, const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for the given dataset.

Parameters:
dataset The dataset to set the attributes for.
a The attributes to set.

Definition at line 406 of file KDChartAbstractDiagram.cpp.

References d.

```
413 {
```

7.44.2.70 void AbstractDiagram::setDataValueAttributes (const QModelIndex & *index*, const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for the given index.

Parameters:
index The datapoint to set the attributes for.
a The attributes to set.

Definition at line 395 of file KDChartAbstractDiagram.cpp.

References d, KDChart::DataValueLabelAttributesRole, and KDChart::AbstractDiagram::propertiesChanged().

```
395 {
396     d->attributesModel->setData(
397         d->attributesModel->mapFromSource( index ),
398         qVariantFromValue( a ),
399         DataValueLabelAttributesRole );
400     emit propertiesChanged();
401 }
402
403
```

7.44.2.71 void AbstractPieDiagram::setGranularity (qreal *value*) `[inherited]`

Set the granularity: the smaller the granularity the more your diagram segments will show facettes instead of rounded segments.

Parameters:

value the granularity value between 0.05 (one twentieth of a degree) and 36.0 (one tenth of a full circle), other values will be interpreted as 1.0.

Definition at line 64 of file KDChartAbstractPieDiagram.cpp.

References d.

```
65 {
66     d->granularity = value;
67 }
```

7.44.2.72 void AbstractDiagram::setHidden (bool *hidden*) `[inherited]`

Hide (or unhide, resp.) all datapoints in the model.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

hidden The hidden status to set.

Definition at line 365 of file KDChartAbstractDiagram.cpp.

References d.

```
372 {
```

7.44.2.73 void AbstractDiagram::setHidden (int *column*, bool *hidden*) `[inherited]`

Hide (or unhide, resp.) a dataset.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

dataset The dataset to set the hidden status for.

hidden The hidden status to set.

Definition at line 356 of file KDChartAbstractDiagram.cpp.

References d.

```
364 {
```

7.44.2.74 void AbstractDiagram::setHidden (const QModelIndex & *index*, bool *hidden*) `[inherited]`

Hide (or unhide, resp.) a data cell.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

index The datapoint to set the hidden status for.

hidden The hidden status to set.

Definition at line 347 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::DataHiddenRole.

```
355 {
```

7.44.2.75 void AbstractDiagram::setModel (QAbstractItemModel $*$ *model*) `[virtual, inherited]`

Associate a model with the diagram.

Definition at line 245 of file KDChartAbstractDiagram.cpp.

References d, KDChart::AttributesModel::initFrom(), and KDChart::AbstractDiagram::modelsChanged().

```
246 {
247   QAbstractItemView::setModel( newModel );
248   AttributesModel* amodel = new PrivateAttributesModel( newModel, this );
249   amodel->initFrom( d->attributesModel );
250   d->setAttributesModel(amodel);
251   scheduleDelayedItemsLayout();
252   d->databoundariesDirty = true;
253   emit modelsChanged();
254 }
```

7.44.2.76 void AbstractDiagram::setPen (const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting all datasets in the model.

Parameters:

pen The pen to use.

Definition at line 740 of file KDChartAbstractDiagram.cpp.

```
746 {
```

7.44.2.77 void AbstractDiagram::setPen (int *dataset*, const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting the given dataset.

Parameters:

dataset The dataset's row in the model.

pen The pen to use.

Definition at line 747 of file KDChartAbstractDiagram.cpp.

```
755 {
```

7.44.2.78 void AbstractDiagram::setPen (const QModelIndex & *index*, const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting the datapoint at the given index.

Parameters:

index The datapoint's index in the model.

pen The pen to use.

Definition at line 732 of file KDChartAbstractDiagram.cpp.

```
739 {
```

7.44.2.79 void AbstractDiagram::setPercentMode (bool *percent*) `[inherited]`

Definition at line 462 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::setType(), and KDChart::BarDiagram::setType().

```
{
```

7.44.2.80 void AbstractPieDiagram::setPieAttributes (int *column*, const PieAttributes & *a*) `[inherited]`

Definition at line 94 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::PieAttributesRole.

```
{
    d->attributesModel->setHeaderData(
        column, Qt::Vertical, qVariantFromValue( attrs ), PieAttributesRole );
    emit layoutChanged( this );
}
```

7.44.2.81 void AbstractPieDiagram::setPieAttributes (const PieAttributes & *a*) `[inherited]`

Definition at line 88 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::PieAttributesRole.

```
{
    d->attributesModel->setModelData( qVariantFromValue( attrs ), PieAttributesRole );
    emit layoutChanged( this );
}
```

7.44.2.82 void AbstractDiagram::setRootIndex (const QModelIndex & *idx*) `[virtual, inherited]`

Set the root index in the model, where the diagram starts referencing data for display.

[reimplemented]

Definition at line 294 of file KDChartAbstractDiagram.cpp.

References d.

7.44.2.83 void AbstractDiagram::setSelection (const QRect & *rect*, QItemSelectionModel::SelectionFlags *command*) `[virtual, inherited]`

[reimplemented]

Definition at line 848 of file KDChartAbstractDiagram.cpp.

```
{ return QRegion(); }
```

7.44.2.84 void AbstractPieDiagram::setStartPosition (int *degrees*) `[inherited]`

Deprecated

Use PolarCoordinatePlane::setStartPosition(qreal degrees) instead.

Definition at line 77 of file KDChartAbstractPieDiagram.cpp.

```
78 {
79     qWarning() << "Deprecated AbstractPieDiagram::setStartPosition() called, setting ignored.";
80 }
```

7.44.2.85 void AbstractPieDiagram::setThreeDPieAttributes (const QModelIndex & *index*, const ThreeDPieAttributes & *a*) `[inherited]`

Definition at line 143 of file KDChartAbstractPieDiagram.cpp.

References KDChart::ThreeDPieAttributesRole.

```
144 {
145     model()->setData( index, qVariantFromValue( tda ), ThreeDPieAttributesRole );
146     emit layoutChanged( this );
147 }
```

7.44.2.86 void AbstractPieDiagram::setThreeDPieAttributes (int *column*, const ThreeDPieAttributes & *a*) `[inherited]`

Definition at line 136 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::ThreeDPieAttributesRole.

```
137 {
138     d->attributesModel->setHeaderData(
139         column, Qt::Vertical, qVariantFromValue( tda ), ThreeDPieAttributesRole );
140     emit layoutChanged( this );
141 }
```

7.44.2.87 void AbstractPieDiagram::setThreeDPieAttributes (const ThreeDPieAttributes & *a*) `[inherited]`

Definition at line 130 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::ThreeDPieAttributesRole.

```
131 {
132     d->attributesModel->setModelData( qVariantFromValue( tda ), ThreeDPieAttributesRole );
133     emit layoutChanged( this );
134 }
```

7.44.2.88 int AbstractPieDiagram::startPosition () const `[inherited]`

Deprecated

Use qreal PolarCoordinatePlane::startPosition instead.

Definition at line 82 of file KDChartAbstractPieDiagram.cpp.

```
{
    qWarning() << "Deprecated AbstractPieDiagram::startPosition() called.";
    return 0;
}
```

7.44.2.89 ThreeDPieAttributes AbstractPieDiagram::threeDPieAttributes (const QModelIndex & *index*) const `[inherited]`

Definition at line 169 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::ThreeDPieAttributesRole.

```
{
    return qVariantValue<ThreeDPieAttributes>(
        d->attributesModel->data(
            d->attributesModel->mapFromSource( index ),
            ThreeDPieAttributesRole ) );
}
```

7.44.2.90 ThreeDPieAttributes AbstractPieDiagram::threeDPieAttributes (int *column*) const `[inherited]`

Definition at line 161 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::ThreeDPieAttributesRole.

```
{
    return qVariantValue<ThreeDPieAttributes>(
        d->attributesModel->data(
            d->attributesModel->mapFromSource( columnToIndex( column ) ).column(),
            ThreeDPieAttributesRole ) );
}
```

7.44.2.91 ThreeDPieAttributes AbstractPieDiagram::threeDPieAttributes () const `[inherited]`

Definition at line 152 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::ThreeDPieAttributesRole.

Referenced by paint().

```
{
    return qVariantValue<ThreeDPieAttributes>(
        d->attributesModel->data( ThreeDPieAttributesRole ) );
}
```

7.44.2.92 void AbstractDiagram::update () const `[inherited]`

Definition at line 961 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractDiagram::doItemsLayout().

7.44.2.93 void KDChart::AbstractDiagram::useDefaultColors () `[inherited]`

Set the palette to be used, for painting datasets to the default palette.

See also:

KDChart::Palette. FIXME: fold into one usePalette (KDChart::Palette&) method

Definition at line 855 of file KDChartAbstractDiagram.cpp.

References d.

```
859 {
```

7.44.2.94 void KDChart::AbstractDiagram::useRainbowColors () `[inherited]`

Set the palette to be used, for painting datasets to the rainbow palette.

See also:

KDChart::Palette.

Definition at line 865 of file KDChartAbstractDiagram.cpp.

References d.

```
869 {
```

7.44.2.95 bool AbstractDiagram::usesExternalAttributesModel () const `[virtual, inherited]`

Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

See also:

setAttributesModel

Definition at line 280 of file KDChartAbstractDiagram.cpp.

References d.

```
281 {
282     return d->usesExternalAttributesModel();
283 }
```

7.44.2.96 void KDChart::AbstractDiagram::useSubduedColors () `[inherited]`

Set the palette to be used, for painting datasets to the subdued palette.

See also:
KDChart::Palette.

Definition at line 860 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.44.2.97 double AbstractDiagram::valueForCell (int *row*, int *column*) const `[protected, inherited]`

Helper method, retrieving the data value (DisplayRole) for a given row and column.

Parameters:

row The row to query.

column The column to query.

Returns:
The value of the display role at the given row and column as a double.

Definition at line 955 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::attributesModelRootIndex(), and d.

Referenced by KDChart::LineDiagram::paint().

```
{
```

7.44.2.98 double PieDiagram::valueTotals () const `[virtual]`

[reimplemented]

Implements KDChart::AbstractPolarDiagram.

Definition at line 1089 of file KDChartPieDiagram.cpp.

References KDChart::AbstractPolarDiagram::columnCount().

Referenced by paint().

```
{
    const int colCount = columnCount();
    double total = 0.0;
    for ( int j = 0; j < colCount; ++j ) {
      total += qAbs(model()->data( model()->index( 0, j, rootIndex() ) ).toDouble());
      //qDebug() << model()->data( model()->index( 0, j, rootIndex() ) ).toDouble();
    }
    return total;
}
```

7.44.2.99 int AbstractDiagram::verticalOffset () const `[virtual, inherited]`

[reimplemented]

Definition at line 842 of file KDChartAbstractDiagram.cpp.

```
844 { return true; }
```

7.44.2.100 QRect AbstractDiagram::visualRect (const QModelIndex & *index*) const `[virtual, inherited]`

[reimplemented]

Definition at line 825 of file KDChartAbstractDiagram.cpp.

```
829 {}
```

7.44.2.101 QRegion AbstractDiagram::visualRegionForSelection (const QItemSelection & *selection*) const `[virtual, inherited]`

[reimplemented]

Definition at line 851 of file KDChartAbstractDiagram.cpp.

7.44.3 Member Data Documentation

7.44.3.1 Q_SIGNALS KDChart::AbstractDiagram::__pad0__ `[protected, inherited]`

Definition at line 589 of file KDChartAbstractDiagram.h.

The documentation for this class was generated from the following files:

- KDChartPieDiagram.h
- KDChartPieDiagram.cpp

7.45 KDChart::PolarCoordinatePlane Class Reference

`#include <KDChartPolarCoordinatePlane.h>`

Inheritance diagram for KDChart::PolarCoordinatePlane:Collaboration diagram for KDChart::PolarCoordinatePlane:

Public Types

- enum AxesCalcMode {
  Linear,
  Logarithmic }
- typedef QList< CoordinateTransformation > CoordinateTransformationList

Public Member Functions

- void addDiagram (AbstractDiagram ∗diagram)

  Adds a diagram to this coordinate plane.

- void alignToReferencePoint (const RelativePosition &position)
- qreal angleUnit () const
- BackgroundAttributes backgroundAttributes () const
- virtual int bottomOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart:AutoSpacerLayoutItem::sizeHint().

- bool compare (const AbstractAreaBase ∗other) const

  Returns true if both areas have the same settings.

- AbstractDiagram ∗ diagram ()
- ConstAbstractDiagramList diagrams () const
- AbstractDiagramList diagrams ()
- virtual Qt::Orientations expandingDirections () const

  pure virtual in QLayoutItem

- FrameAttributes frameAttributes () const
- virtual QRect geometry () const

  pure virtual in QLayoutItem

- void getFrameLeadings (int &left, int &top, int &right, int &bottom) const
- GridAttributes globalGridAttributes () const
- const GridAttributes gridAttributes (bool circular) const
- DataDimensionsList gridDimensionsList ()

  Returns the dimensions used for drawing the grid lines.

- bool hasOwnGridAttributes (bool circular) const
- virtual bool isEmpty () const

  pure virtual in QLayoutItem

- const bool isVisiblePoint (const QPointF &point) const

Tests, if a point is visible on the coordinate plane.

- void layoutPlanes ()

  Calling layoutPlanes() on the plane triggers the global KDChart::Chart::slotLayoutPlanes().

- virtual int leftOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- virtual QSize maximumSize () const

  pure virtual in QLayoutItem

- virtual QSize minimumSize () const

  pure virtual in QLayoutItem

- virtual QSize minimumSizeHint () const

  [reimplemented]

- void mousePressEvent (QMouseEvent ∗event)

  reimp

- void needLayoutPlanes ()

  Emitted when plane needs to trigger the Chart's layouting of the coord.

- void needRelayout ()

  Emitted when plane needs to trigger the Chart's layouting.

- void needUpdate ()

  Emitted when plane needs to update its drawings.

- virtual void paint (QPainter ∗)

  reimpl

- virtual void paintAll (QPainter &painter)

  Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

- virtual void paintBackground (QPainter &painter, const QRect &rectangle)
- virtual void paintCtx (PaintContext ∗context)

  Default impl: Paint the complete item using its layouted position and size.

- virtual void paintFrame (QPainter &painter, const QRect &rectangle)
- virtual void paintIntoRect (QPainter &painter, const QRect &rect)

  Draws the background and frame, then calls paint().

- const Chart ∗ parent () const
- Chart ∗ parent ()
- QLayout ∗ parentLayout ()
- PolarCoordinatePlane (Chart ∗parent=0)
- void propertiesChanged ()

  Emitted upon change of a property of the Coordinate Plane or any of its components.

- qreal radiusUnit () const
- AbstractCoordinatePlane ∗ referenceCoordinatePlane () const

  There are two ways, in which planes can be caused to interact, in where they are put layouting wise: The first is the reference plane.

- void relayout ()

  Calling relayout() on the plane triggers the global KDChart::Chart::slotRelayout().

- void removeFromParentLayout ()
- virtual void replaceDiagram (AbstractDiagram ∗diagram, AbstractDiagram ∗oldDiagram=0)

  Replaces the old diagram, or appends the diagram, it there is none yet.

- void resetGridAttributes (bool circular)

  Reset the attributes to be used for grid lines drawn in circular direction (or in sagittal direction, resp.).

- virtual int rightOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- void setBackgroundAttributes (const BackgroundAttributes &a)
- void setFrameAttributes (const FrameAttributes &a)
- virtual void setGeometry (const QRect &r)

  pure virtual in QLayoutItem

- void setGlobalGridAttributes (const GridAttributes &)

  Set the grid attributes to be used by this coordinate plane.

- void setGridAttributes (bool circular, const GridAttributes &)

  Set the attributes to be used for grid lines drawn in circular direction (or in sagittal direction, resp.).

- void setGridNeedsRecalculate ()

  Used by the chart to clear the cached grid data.

- void setParent (Chart ∗parent)

  Called internally by KDChart::Chart.

- void setParentLayout (QLayout ∗lay)
- virtual void setParentWidget (QWidget ∗widget)

  Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

- void setReferenceCoordinatePlane (AbstractCoordinatePlane ∗plane)

  Set another coordinate plane to be used as the reference plane for this one.

- void setStartPosition (qreal degrees)

  Specify the rotation of the coordinate plane.

- virtual void setZoomCenter (QPointF center)

  Set the point (in value coordinates) to be used as the center point in zoom operations.

- virtual void setZoomFactorX (double factor)

  Sets the zoom factor in horizontal direction, that is applied to all coordinate transformations.

- virtual void setZoomFactorY (double factor)

  Sets the zoom factor in vertical direction, that is applied to all coordinate transformations.

- virtual QSize sizeHint () const

  pure virtual in QLayoutItem

- virtual void sizeHintChanged () const

  Report changed size hint: ask the parent widget to recalculate the layout.

- virtual QSizePolicy sizePolicy () const

  [reimplemented]

- qreal startPosition () const

  Retrieve the rotation of the coordinate plane.

- virtual void takeDiagram (AbstractDiagram ∗diagram)

  Removes the diagram from the plane, without deleting it.

- virtual int topOverlap (bool doNotRecalculate=false) const

  This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

- const QPointF translate (const QPointF &diagramPoint) const

  Translate the given point in value space coordinates to a position in pixel space.

- const QPointF translatePolar (const QPointF &diagramPoint) const
- virtual QPointF zoomCenter () const
- virtual double zoomFactorX () const
- virtual double zoomFactorY () const
- ∼PolarCoordinatePlane ()

Static Public Member Functions

- void paintBackgroundAttributes (QPainter &painter, const QRect &rectangle, const KDChart::BackgroundAttributes &attributes)
- void paintFrameAttributes (QPainter &painter, const QRect &rectangle, const KDChart::FrameAttributes &attributes)

Public Attributes

- Q_SIGNALS __pad0__: void destroyedCoordinatePlane(AbstractCoordinatePlane∗)

Protected Member Functions

- virtual QRect areaGeometry () const
- virtual DataDimensionsList getDataDimensionsList () const
- QRect innerRect () const
- void layoutDiagrams ()

  Distribute the available space among the diagrams and axes.

- void paintEvent (QPaintEvent ∗)
- virtual void positionHasChanged ()
- void resizeEvent (QResizeEvent ∗)

Protected Attributes

- QWidget ∗ mParent
- QLayout ∗ mParentLayout
- protected Q_SLOTS: void slotLayoutChanged(AbstractDiagram∗ diagram)

7.45.1 Member Typedef Documentation

7.45.1.1 typedef QList<CoordinateTransformation> KDChart::PolarCoordinatePlane::CoordinateTransformationList

Definition at line 45 of file KDChartPolarCoordinatePlane.h.

7.45.2 Member Enumeration Documentation

7.45.2.1 enum KDChart::AbstractCoordinatePlane::AxesCalcMode `[inherited]`

Enumeration values:

Linear

Logarithmic

Definition at line 55 of file KDChartAbstractCoordinatePlane.h.

```
{ Linear, Logarithmic };
```

7.45.3 Constructor & Destructor Documentation

7.45.3.1 PolarCoordinatePlane::PolarCoordinatePlane (Chart ∗ *parent* = 0) `[explicit]`

Definition at line 114 of file KDChartPolarCoordinatePlane.cpp.

```
    : AbstractCoordinatePlane ( new Private(), parent )
{
    // this bloc left empty intentionally
}
```

7.45.3.2 PolarCoordinatePlane::∼PolarCoordinatePlane ()

Definition at line 120 of file KDChartPolarCoordinatePlane.cpp.

```
121 {
122     // this bloc left empty intentionally
123 }
```

7.45.4 Member Function Documentation

7.45.4.1 void PolarCoordinatePlane::addDiagram (AbstractDiagram ∗ *diagram*) `[virtual]`

Adds a diagram to this coordinate plane.

Parameters:

diagram The diagram to add.

See also:

replaceDiagram, takeDiagram

Reimplemented from KDChart::AbstractCoordinatePlane.

Definition at line 130 of file KDChartPolarCoordinatePlane.cpp.

References KDChart::AbstractCoordinatePlane::addDiagram().

```
131 {
132     Q_ASSERT_X ( dynamic_cast<AbstractPolarDiagram*> ( diagram ),
133                  "PolarCoordinatePlane::addDiagram", "Only polar"
134                  "diagrams can be added to a polar coordinate plane!" );
135     AbstractCoordinatePlane::addDiagram ( diagram );
136     connect ( diagram,  SIGNAL ( layoutChanged ( AbstractDiagram* ) ),
137               SLOT ( slotLayoutChanged ( AbstractDiagram* ) ) );
138
139 }
```

7.45.4.2 void AbstractAreaBase::alignToReferencePoint (const RelativePosition & *position*) `[inherited]`

Definition at line 90 of file KDChartAbstractAreaBase.cpp.

```
91 {
92     Q_UNUSED( position );
93     // PENDING(kalle) FIXME
94     qWarning( "Sorry, not implemented: void AbstractAreaBase::alignToReferencePoint( const RelativePosi
95 }
```

7.45.4.3 qreal PolarCoordinatePlane::angleUnit () const

Definition at line 294 of file KDChartPolarCoordinatePlane.cpp.

References d.

Referenced by layoutDiagrams().

```
{
    Q_ASSERT_X ( d->currentTransformation != 0, "PolarCoordinatePlane::angleUnit",
                 "Only call angleUnit() from within paint()." );
    return  d->currentTransformation->angleUnit;
}
```

7.45.4.4 QRect AbstractArea::areaGeometry () const `[protected, virtual, inherited]`

Implements KDChart::AbstractAreaBase.

Definition at line 150 of file KDChartAbstractArea.cpp.

Referenced by KDChart::CartesianCoordinatePlane::drawingArea(), layoutDiagrams(), KDChart::CartesianAxis::paint(), KDChart::AbstractArea::paintAll(), and KDChart::CartesianAxis::paintCtx().

```
{
    return geometry();
}
```

7.45.4.5 BackgroundAttributes AbstractAreaBase::backgroundAttributes () const `[inherited]`

Definition at line 112 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by updateCommonBrush().

```
{
    return d->backgroundAttributes;
}
```

7.45.4.6 int AbstractArea::bottomOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart:AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the bottom edge of the area.

Note:

> The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 101 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
102 {
103     // Re-calculate the sizes,
104     // so we also get the amountOf..Overlap members set newly:
105     if( ! doNotRecalculate )
106         sizeHint();
107     return d->amountOfBottomOverlap;
108 }
```

7.45.4.7 bool AbstractAreaBase::compare (const AbstractAreaBase ∗ *other*) const `[inherited]`

Returns true if both areas have the same settings.

Definition at line 75 of file KDChartAbstractAreaBase.cpp.

```
76 {
77     if( other == this ) return true;
78     if( ! other ){
79         //qDebug() << "CartesianAxis::compare() cannot compare to Null pointer";
80         return false;
81     }
82     /*
83     qDebug() << "AbstractAreaBase:" << (frameAttributes() == other->frameAttributes())
84         << (backgroundAttributes() == other->backgroundAttributes()) << "\n";
85     */
86     return  (frameAttributes()      == other->frameAttributes()) &&
87             (backgroundAttributes() == other->backgroundAttributes());
88 }
```

7.45.4.8 AbstractDiagram ∗ AbstractCoordinatePlane::diagram () `[inherited]`

Returns:

The first diagram associated with this coordinate plane.

Definition at line 113 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::Widget::diagram(), KDChart::Chart::mousePressEvent(), and setStartPosition().

```
114 {
115     if ( d->diagrams.isEmpty() )
116     {
117         return 0;
118     } else {
119         return d->diagrams.first();
120     }
121 }
```

7.45.4.9 ConstAbstractDiagramList AbstractCoordinatePlane::diagrams () const `[inherited]`

Returns:

The list of diagrams associated with this coordinate plane.

Definition at line 128 of file KDChartAbstractCoordinatePlane.cpp.

References KDChart::ConstAbstractDiagramList, and d.

```
129 {
130     ConstAbstractDiagramList list;
131 #ifndef QT_NO_STL
132     qCopy( d->diagrams.begin(), d->diagrams.end(), std::back_inserter( list ) );
133 #else
134     Q_FOREACH( AbstractDiagram * a, d->diagrams )
135         list.push_back( a );
136 #endif
137     return list;
138 }
```

7.45.4.10 AbstractDiagramList AbstractCoordinatePlane::diagrams () `[inherited]`

Returns:

The list of diagrams associated with this coordinate plane.

Definition at line 123 of file KDChartAbstractCoordinatePlane.cpp.

References KDChart::AbstractDiagramList, and d.

Referenced by KDChart::CartesianCoordinatePlane::getDataDimensionsList(), KDChart::CartesianCoordinatePlane::getRawDataBoundingRectFromDiagrams(), layoutDiagrams(), KDChart::CartesianCoordinatePlane::layoutDiagrams(), KDChart::Chart::mousePressEvent(), paint(), and KDChart::CartesianCoordinatePlane::paint().

```
124 {
125     return d->diagrams;
126 }
```

7.45.4.11 Qt::Orientations KDChart::AbstractCoordinatePlane::expandingDirections () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 208 of file KDChartAbstractCoordinatePlane.cpp.

```
209 {
210     return Qt::Vertical | Qt::Horizontal;
211 }
```

7.45.4.12 FrameAttributes AbstractAreaBase::frameAttributes () const `[inherited]`

Definition at line 102 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone(), and updateCommonBrush().

```
103 {
104     return d->frameAttributes;
105 }
```

7.45.4.13 QRect KDChart::AbstractCoordinatePlane::geometry () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 242 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::Chart::mousePressEvent(), and paint().

```
{
    return d->geometry;
}
```

7.45.4.14 DataDimensionsList PolarCoordinatePlane::getDataDimensionsList () const `[protected, virtual]`

Implements KDChart::AbstractCoordinatePlane.

Definition at line 358 of file KDChartPolarCoordinatePlane.cpp.

References KDChart::DataDimensionsList.

```
{
    DataDimensionsList l;

    //FIXME(khz): do the real calculation

    return l;
}
```

7.45.4.15 void AbstractAreaBase::getFrameLeadings (int & *left*, int & *top*, int & *right*, int & *bottom*) const `[inherited]`

Definition at line 204 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::AbstractAreaBase::innerRect(), and KDChart::AbstractAreaWidget::paintAll().

```
{
    if( d && d->frameAttributes.isVisible() ){
        const int padding = qMax( d->frameAttributes.padding(), 0 );
        left   = padding;
        top    = padding;
        right  = padding;
        bottom = padding;
    }else{
        left   = 0;
        top    = 0;
        right  = 0;
        bottom = 0;
    }
}
```

7.45.4.16 GridAttributes KDChart::AbstractCoordinatePlane::globalGridAttributes () const `[inherited]`

Returns:

The grid attributes used by this coordinate plane.

See also:

setGlobalGridAttributes
CartesianCoordinatePlane::gridAttributes

Definition at line 157 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by gridAttributes(), and KDChart::CartesianCoordinatePlane::gridAttributes().

```
{
    return d->gridAttributes;
}
```

7.45.4.17 const GridAttributes KDChart::PolarCoordinatePlane::gridAttributes (bool *circular*) const

Returns:

The attributes used for grid lines drawn in circular direction (or in sagittal direction, resp.).

Note:

This function always returns a valid set of grid attributes: If no special grid attributes were set for this direction the global attributes are returned, as returned by AbstractCoordinatePlane::globalGridAttributes.

See also:

setGridAttributes
resetGridAttributes
AbstractCoordinatePlane::globalGridAttributes
hasOwnGridAttributes

Definition at line 387 of file KDChartPolarCoordinatePlane.cpp.

References d, KDChart::AbstractCoordinatePlane::globalGridAttributes(), and hasOwnGridAttributes().

```
{
    if( hasOwnGridAttributes( circular ) ){
        if( circular )
            return d->gridAttributesCircular;
        else
            return d->gridAttributesSagittal;
    }else{
        return globalGridAttributes();
    }
}
```

7.45.4.18 KDChart::DataDimensionsList KDChart::AbstractCoordinatePlane::gridDimensionsList () `[inherited]`

Returns the dimensions used for drawing the grid lines.

Returned data is the result of (cached) grid calculations, so - if you need that information for your own tasks - make sure to call again this function after every data modification that has changed the data range, since grid calculation is based upon the data range, thus the grid start/end might have changed if the data was changed.

Note:

Returned list will contain different numbers of DataDimension, depending on the kind of coordinate plane used. For CartesianCoordinatePlane two DataDimension are returned: the first representing grid lines in X direction (matching the Abscissa axes) and the second indicating vertical grid lines (or Ordinate axes, resp.).

Returns:

The dimensions used for drawing the grid lines.

See also:

DataDimension

Definition at line 162 of file KDChartAbstractCoordinatePlane.cpp.

References d, and KDChart::DataDimensionsList.

Referenced by KDChart::CartesianCoordinatePlane::layoutDiagrams(), KDChart::CartesianAxis::maximumSize(), and KDChart::CartesianAxis::paintCtx().

```
163 {
164     //KDChart::DataDimensionsList l( d->grid->updateData( this ) );
165     //qDebug() << "AbstractCoordinatePlane::gridDimensionsList() Y-range:" << l.last().end - l.last().
166     //qDebug() << "AbstractCoordinatePlane::gridDimensionsList() X-range:" << l.first().end - l.first(
167     return d->grid->updateData( this );
168 }
```

7.45.4.19 bool KDChart::PolarCoordinatePlane::hasOwnGridAttributes (bool *circular*) const

Returns:

Returns whether the grid attributes have been set for the respective direction via setGridAttributes(bool circular).

If false, the grid will use the global attributes set by AbstractCoordinatePlane::globalGridAttributes (or the default attributes, resp.)

See also:

setGridAttributes
resetGridAttributes
AbstractCoordinatePlane::globalGridAttributes

Definition at line 410 of file KDChartPolarCoordinatePlane.cpp.

References d.

Referenced by gridAttributes().

```
{
    return
        ( circular )
        ? d->hasOwnGridAttributesCircular
        : d->hasOwnGridAttributesSagittal;
}
```

7.45.4.20 QRect AbstractAreaBase::innerRect () const `[protected, inherited]`

Definition at line 220 of file KDChartAbstractAreaBase.cpp.

References KDChart::AbstractAreaBase::areaGeometry(), and KDChart::AbstractAreaBase::getFrameLeadings().

Referenced by KDChart::TextArea::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    int left;
    int top;
    int right;
    int bottom;
    getFrameLeadings( left, top, right, bottom );
    return
        QRect( QPoint(0,0), areaGeometry().size() )
            .adjusted( left, top, -right, -bottom );
}
```

7.45.4.21 bool KDChart::AbstractCoordinatePlane::isEmpty () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 201 of file KDChartAbstractCoordinatePlane.cpp.

```
{
    return false; // never empty!
    // coordinate planes with no associated diagrams
    // are showing a default grid of ()1..10, 1..10) stepWidth 1
}
```

7.45.4.22 const bool KDChart::AbstractCoordinatePlane::isVisiblePoint (const QPointF & *point*) const `[inherited]`

Tests, if a point is visible on the coordinate plane.

Note:

Before calling this function the point must have been translated into coordinate plane space.

Definition at line 275 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
{
    return d->isVisiblePoint( this, point );
}
```

7.45.4.23 void PolarCoordinatePlane::layoutDiagrams () `[protected, virtual]`

Distribute the available space among the diagrams and axes.

Implements KDChart::AbstractCoordinatePlane.

Definition at line 238 of file KDChartPolarCoordinatePlane.cpp.

References angleUnit(), KDChart::AbstractArea::areaGeometry(), d, KDChart::AbstractDiagram::dataBoundaries(), KDChart::AbstractCoordinatePlane::diagrams(), radiusUnit(), startPosition(), and KDChart::AbstractPolarDiagram::valueTotals().

Referenced by resizeEvent().

```
{
    // the rectangle the diagrams cover in the *plane*:
    // (Why -3? We save 1px on each side for the antialiased drawing, and
    // respect the way QPainter calculates the width of a painted rect (the
    // size is the rectangle size plus the pen width). This way, most clipping
    // for regular pens should be avoided. When pens with a penWidth or larger
    // than 1 are used, this may not b sufficient.
    const QRect rect( areaGeometry() );
    d->contentRect = QRectF ( 1, 1, rect.width() - 3, rect.height() - 3 );

    // FIXME distribute space according to options:
    const qreal oldStartPosition = startPosition();
    d->coordinateTransformations.clear();
    Q_FOREACH( AbstractDiagram* diagram, diagrams() )
        {
            AbstractPolarDiagram *polarDiagram = dynamic_cast<AbstractPolarDiagram*>( diagram );
            Q_ASSERT( polarDiagram );
            QPair<QPointF, QPointF> dataBoundariesPair = polarDiagram->dataBoundaries();

            const double angleUnit = 360 / polarDiagram->valueTotals();
//qDebug() << "--------------------------------------------------------";
            const double radius = dataBoundariesPair.second.y();
//qDebug() << radius <<"="<<dataBoundariesPair.second.y();
            const double diagramWidth = radius * 2; // == height
            const double planeWidth = d->contentRect.width();
            const double planeHeight = d->contentRect.height();
            const double radiusUnit = qMin( planeWidth, planeHeight ) / diagramWidth;
//qDebug() << radiusUnit <<"=" << "qMin( "<<planeWidth<<","<< planeHeight <<") / "<<diagramWidth;
            QPointF coordinateOrigin = QPointF ( planeWidth / 2, planeHeight / 2 );
            coordinateOrigin += d->contentRect.topLeft();

            CoordinateTransformation diagramTransposition;
            diagramTransposition.originTranslation = coordinateOrigin;
            diagramTransposition.radiusUnit = radiusUnit;
            diagramTransposition.angleUnit = angleUnit;
            diagramTransposition.startPosition = oldStartPosition;
            diagramTransposition.zoom = ZoomParameters();
            d->coordinateTransformations.append( diagramTransposition );
        }
}
```

7.45.4.24 void KDChart::AbstractCoordinatePlane::layoutPlanes () `[inherited]`

Calling layoutPlanes() on the plane triggers the global KDChart::Chart::slotLayoutPlanes().

Definition at line 259 of file KDChartAbstractCoordinatePlane.cpp.

References KDChart::AbstractCoordinatePlane::needLayoutPlanes().

Referenced by KDChart::AbstractCoordinatePlane::addDiagram(), KDChart::CartesianAxis::layout-

Planes(), KDChart::AbstractCartesianDiagram::layoutPlanes(), and KDChart::AbstractCoordinatePlane::replaceDiagram().

```
{
    //qDebug("KDChart::AbstractCoordinatePlane::relayout() called");
    emit needLayoutPlanes();
}
```

7.45.4.25 int AbstractArea::leftOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the left edge of the area.

Note:

The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 77 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
    // Re-calculate the sizes,
    // so we also get the amountOf..Overlap members set newly:
    if( ! doNotRecalculate )
        sizeHint();
    return d->amountOfLeftOverlap;
}
```

7.45.4.26 QSize KDChart::AbstractCoordinatePlane::maximumSize () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 213 of file KDChartAbstractCoordinatePlane.cpp.

Referenced by KDChart::AbstractCoordinatePlane::sizeHint().

```
{
    // No maximum size set. Especially not parent()->size(), we are not layouting
    // to the parent widget's size when using Chart::paint()!
    return QSize(QLAYOUTSIZE_MAX, QLAYOUTSIZE_MAX);
}
```

7.45.4.27 QSize KDChart::AbstractCoordinatePlane::minimumSize () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 220 of file KDChartAbstractCoordinatePlane.cpp.

```
221 {
222     return QSize(60, 60); // this default can be overwritten by derived classes
223 }
```

7.45.4.28 QSize KDChart::AbstractCoordinatePlane::minimumSizeHint () const `[virtual, inherited]`

[reimplemented]

Definition at line 140 of file KDChartAbstractCoordinatePlane.cpp.

```
141 {
142     return QSize( 200, 200 );
143 }
```

7.45.4.29 void KDChart::AbstractCoordinatePlane::mousePressEvent (QMouseEvent ∗ *event*) `[inherited]`

reimp

Definition at line 266 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::Chart::mousePressEvent().

```
267 {
268     KDAB_FOREACH( AbstractDiagram * a, d->diagrams )
269     {
270         a->mousePressEvent( event );
271     }
272 }
```

7.45.4.30 void KDChart::AbstractCoordinatePlane::needLayoutPlanes () `[inherited]`

Emitted when plane needs to trigger the Chart's layouting of the coord.

planes.

Referenced by KDChart::AbstractCoordinatePlane::layoutPlanes().

7.45.4.31 void KDChart::AbstractCoordinatePlane::needRelayout () `[inherited]`

Emitted when plane needs to trigger the Chart's layouting.

Referenced by KDChart::AbstractCoordinatePlane::relayout().

7.45.4.32 void KDChart::AbstractCoordinatePlane::needUpdate () `[inherited]`

Emitted when plane needs to update its drawings.

7.45.4.33 void PolarCoordinatePlane::paint (QPainter ∗) `[virtual]`

reimpl

Implements KDChart::AbstractLayoutItem.

Definition at line 141 of file KDChartPolarCoordinatePlane.cpp.

References KDChart::AbstractDiagramList, d, KDChart::AbstractCoordinatePlane::diagrams(), KDChart::AbstractCoordinatePlane::geometry(), KDChart::PaintContext::setCoordinatePlane(), KDChart::PaintContext::setPainter(), and KDChart::PaintContext::setRectangle().

```
{
    AbstractDiagramList diags = diagrams();
    if ( d->coordinateTransformations.size() == diags.size() )
    {
        PaintContext ctx;
        ctx.setPainter ( painter );
        ctx.setCoordinatePlane ( this );
        ctx.setRectangle ( geometry() /*d->contentRect*/ );

        // paint the coordinate system rulers:
        d->currentTransformation = & ( d->coordinateTransformations[0] );
        d->grid->drawGrid( &ctx );

        // paint the diagrams:
        for ( int i = 0; i < diags.size(); i++ )
        {
            d->currentTransformation = & ( d->coordinateTransformations[i] );
            PainterSaver painterSaver( painter );
            diags[i]->paint ( &ctx );
        }
        d->currentTransformation = 0;
    } // else: diagrams have not been set up yet
}
```

7.45.4.34 void AbstractArea::paintAll (QPainter & *painter*) `[virtual, inherited]`

Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

Reimplemented from KDChart::AbstractLayoutItem.

Definition at line 123 of file KDChartAbstractArea.cpp.

References KDChart::AbstractArea::areaGeometry(), d, KDChart::AbstractAreaBase::innerRect(), KDChart::AbstractLayoutItem::paint(), KDChart::AbstractAreaBase::paintBackground(), and KDChart::AbstractAreaBase::paintFrame().

Referenced by KDChart::AbstractArea::paintIntoRect().

```
{
    // Paint the background and frame
    const QRect overlappingArea( geometry().adjusted(
            -d->amountOfLeftOverlap,
            -d->amountOfTopOverlap,
            d->amountOfRightOverlap,
            d->amountOfBottomOverlap ) );
    paintBackground( painter, overlappingArea );
    paintFrame(      painter, overlappingArea );

    // temporarily adjust the widget size, to be sure all content gets calculated
    // to fit into the inner rectangle
136     const QRect oldGeometry( areaGeometry()  );
137     QRect inner( innerRect() );
138     inner.moveTo(
139         oldGeometry.left() + inner.left(),
140         oldGeometry.top()  + inner.top() );
141     const bool needAdjustGeometry = oldGeometry != inner;
142     if( needAdjustGeometry )
143         setGeometry( inner );
144     paint( &painter );
145     if( needAdjustGeometry )
146         setGeometry( oldGeometry );
147     //qDebug() << "AbstractAreaWidget::paintAll() done.";
148 }
```

7.45.4.35 void AbstractAreaBase::paintBackground (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 188 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintBackgroundAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
189 {
190     Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintBackground()",
191                 "Private class was not initialized!" );
192     paintBackgroundAttributes( painter, rect, d->backgroundAttributes );
193 }
```

7.45.4.36 void AbstractAreaBase::paintBackgroundAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::BackgroundAttributes & *attributes*) `[static, inherited]`

Definition at line 119 of file KDChartAbstractAreaBase.cpp.

References KDChart::BackgroundAttributes::brush(), KDChart::BackgroundAttributes::isVisible(), KDChart::BackgroundAttributes::pixmap(), and KDChart::BackgroundAttributes::pixmapMode().

Referenced by KDChart::AbstractAreaBase::paintBackground().

```
121 {
122     if( !attributes.isVisible() ) return;
123
124     /* first draw the brush (may contain a pixmap)*/
125     if( Qt::NoBrush != attributes.brush().style() ) {
126         KDChart::PainterSaver painterSaver( &painter );
127         painter.setPen( Qt::NoPen );
128         const QPointF newTopLeft( painter.deviceMatrix().map( rect.topLeft() ) );
129         painter.setBrushOrigin( newTopLeft );
130         painter.setBrush( attributes.brush() );
131         painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
132     }
133     /* next draw the backPixmap over the brush */
134     if( !attributes.pixmap().isNull() &&
135         attributes.pixmapMode() != BackgroundAttributes::BackgroundPixmapModeNone ) {
136         QPointF ol = rect.topLeft();
137         if( BackgroundAttributes::BackgroundPixmapModeCentered == attributes.pixmapMode() )
138         {
139             ol.setX( rect.center().x() - attributes.pixmap().width() / 2 );
```

```
            ol.setY( rect.center().y() - attributes.pixmap().height()/ 2 );
            painter.drawPixmap( ol, attributes.pixmap() );
        } else {
            QMatrix m;
            double zW = (double)rect.width()  / (double)attributes.pixmap().width();
            double zH = (double)rect.height() / (double)attributes.pixmap().height();
            switch( attributes.pixmapMode() ) {
            case BackgroundAttributes::BackgroundPixmapModeScaled:
            {
                double z;
                z = qMin( zW, zH );
                m.scale( z, z );
            }
            break;
            case BackgroundAttributes::BackgroundPixmapModeStretched:
                m.scale( zW, zH );
                break;
            default:
                ; // Cannot happen, previously checked
            }
            QPixmap pm = attributes.pixmap().transformed( m );
            ol.setX( rect.center().x() - pm.width() / 2 );
            ol.setY( rect.center().y() - pm.height()/ 2 );
            painter.drawPixmap( ol, pm );
        }
    }
}
```

7.45.4.37 void KDChart::AbstractLayoutItem::paintCtx (PaintContext ∗ *context*) `[virtual, inherited]`

Default impl: Paint the complete item using its layouted position and size.

Reimplemented in KDChart::CartesianAxis.

Definition at line 77 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint(), and KDChart::PaintContext::painter().

```
{
    if( context )
        paint( context->painter() );
}
```

7.45.4.38 void KDChart::PolarCoordinatePlane::paintEvent (QPaintEvent ∗) `[protected]`

7.45.4.39 void AbstractAreaBase::paintFrame (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 196 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintFrameAttributes().

Referenced by KDChart::TextArea::paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintFrame()",
                "Private class was not initialized!" );
    paintFrameAttributes( painter, rect, d->frameAttributes );
}
```

7.45.4.40 void AbstractAreaBase::paintFrameAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::FrameAttributes & *attributes*) `[static, inherited]`

Definition at line 169 of file KDChartAbstractAreaBase.cpp.

References KDChart::FrameAttributes::isVisible(), and KDChart::FrameAttributes::pen().

Referenced by KDChart::AbstractAreaBase::paintFrame().

```
{

    if( !attributes.isVisible() ) return;

    // Note: We set the brush to NoBrush explicitely here.
    //       Otherwise we might get a filled rectangle, so any
    //       previously drawn background would be overwritten by that area.

    const QPen   oldPen(   painter.pen() );
    const QBrush oldBrush( painter.brush() );
    painter.setPen(   attributes.pen() );
    painter.setBrush( Qt::NoBrush );
    painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
    painter.setBrush( oldBrush );
    painter.setPen(   oldPen );
}
```

7.45.4.41 void AbstractArea::paintIntoRect (QPainter & *painter*, const QRect & *rect*) `[virtual, inherited]`

Draws the background and frame, then calls paint().

In most cases there is no need to overwrite this method in a derived class, but you would overwrite AbstractLayoutItem::paint() instead.

Definition at line 111 of file KDChartAbstractArea.cpp.

References KDChart::AbstractArea::paintAll().

```
{
    const QRect oldGeometry( geometry() );
    if( oldGeometry != rect )
        setGeometry( rect );
    painter.translate( rect.left(), rect.top() );
    paintAll( painter );
    painter.translate( -rect.left(), -rect.top() );
    if( oldGeometry != rect )
        setGeometry( oldGeometry );
}
```

7.45.4.42 const KDChart::Chart ∗ KDChart::AbstractCoordinatePlane::parent () const `[inherited]`

Definition at line 190 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
{
    return d->parent;
}
```

7.45.4.43 KDChart::Chart ∗ KDChart::AbstractCoordinatePlane::parent () `[inherited]`

Definition at line 195 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::CartesianAxis::maximumSize(), and KDChart::CartesianAxis::paintCtx().

```
{
    return d->parent;
}
```

7.45.4.44 QLayout∗ KDChart::AbstractLayoutItem::parentLayout () `[inherited]`

Definition at line 74 of file KDChartLayoutItems.h.

```
        {
            return mParentLayout;
        }
```

7.45.4.45 void AbstractArea::positionHasChanged () `[protected, virtual, inherited]`

Reimplemented from KDChart::AbstractAreaBase.

Definition at line 155 of file KDChartAbstractArea.cpp.

```
{
    emit positionChanged( this );
}
```

7.45.4.46 void KDChart::AbstractCoordinatePlane::propertiesChanged () `[inherited]`

Emitted upon change of a property of the Coordinate Plane or any of its components.

Referenced by KDChart::CartesianCoordinatePlane::addDiagram(), KDChart::CartesianCoordinatePlane::adjustHorizontalRangeToData(), KDChart::CartesianCoordinatePlane::adjustVerticalRangeToData(), KDChart::CartesianCoordinatePlane::setAutoAdjustGridToZoom(), KDChart::CartesianCoordinatePlane::setAutoAdjustHorizontalRangeToData(), KDChart::CartesianCoordinatePlane::setAutoAdjustVerticalRangeToData(), KDChart::CartesianCoordinatePlane::setAxesCalcModes(), KDChart::CartesianCoordinatePlane::setAxesCalcModeX(), KDChart::CartesianCoordinatePlane::setAxesCalcModeY(), setGridAttributes(), KDChart::CartesianCoordinatePlane::setGridAttributes(), KDChart::CartesianCoordinatePlane::setHorizontalRange(), KDChart::CartesianCoordinatePlane::setIsometricScaling(), KDChart::CartesianCoordinatePlane::setVerticalRange(), KDChart::CartesianCoordinatePlane::setZoomCenter(), KDChart::CartesianCoordinatePlane::setZoomFactorX(), and KDChart::CartesianCoordinatePlane::setZoomFactorY().

7.45.4.47 qreal PolarCoordinatePlane::radiusUnit () const

Definition at line 301 of file KDChartPolarCoordinatePlane.cpp.

References d.

Referenced by layoutDiagrams().

```
302 {
303     Q_ASSERT_X ( d->currentTransformation != 0, "PolarCoordinatePlane::radiusUnit",
304                  "Only call radiusUnit() from within paint()." );
305     return  d->currentTransformation->radiusUnit;
306 }
```

7.45.4.48 AbstractCoordinatePlane ∗ KDChart::AbstractCoordinatePlane::referenceCoordinatePlane () const `[inherited]`

There are two ways, in which planes can be caused to interact, in where they are put layouting wise: The first is the reference plane.

If such a reference plane is set, on a plane, it will use the same cell in the layout as that one. In addition to this, planes can share an axis. In that case they will be layed out in relation to each other as suggested by the position of the axis. If, for example Plane1 and Plane2 share an axis at position Left, that will result in the layout: Axis Plane1 Plane 2, vertically. If Plane1 also happens to be Plane2's reference plane, both planes are drawn over each other. The reference plane concept allows two planes to share the same space even if neither has any axis, and in case there are shared axis, it is used to decided, whether the planes should be painted on top of each other or layed out vertically or horizontally next to each other.

Returns:

The reference coordinate plane associated with this one.

Definition at line 180 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
181 {
182     return d->referenceCoordinatePlane;
183 }
```

7.45.4.49 void KDChart::AbstractCoordinatePlane::relayout () `[inherited]`

Calling relayout() on the plane triggers the global KDChart::Chart::slotRelayout().

Definition at line 253 of file KDChartAbstractCoordinatePlane.cpp.

References KDChart::AbstractCoordinatePlane::needRelayout().

```
254 {
255     //qDebug("KDChart::AbstractCoordinatePlane::relayout() called");
256     emit needRelayout();
257 }
```

7.45.4.50 void KDChart::AbstractLayoutItem::removeFromParentLayout () `[inherited]`

Definition at line 78 of file KDChartLayoutItems.h.

Referenced by KDChart::Chart::takeCoordinatePlane().

```
79         {
80             if( mParentLayout ){
81                 if( widget() )
82                     mParentLayout->removeWidget( widget() );
```

```
                else
                    mParentLayout->removeItem( this );
            }
        }
```

7.45.4.51 void AbstractCoordinatePlane::replaceDiagram (AbstractDiagram ∗ *diagram*, AbstractDiagram ∗ *oldDiagram* = 0) `[virtual, inherited]`

Replaces the old diagram, or appends the diagram, it there is none yet.

Parameters:

diagram The diagram to be used instead of the old diagram. This parameter must not be zero, or the method will do nothing.

oldDiagram The diagram to be removed by the new diagram. This diagram will be deleted automatically. If the parameter is omitted, the very first diagram will be replaced. In case, there was no diagram yet, the new diagram will just be added.

Note:

If you want to re-use the old diagram, call takeDiagram and addDiagram, instead of using replaceDiagram.

See also:

addDiagram, takeDiagram

Definition at line 82 of file KDChartAbstractCoordinatePlane.cpp.

References KDChart::AbstractCoordinatePlane::addDiagram(), d, KDChart::AbstractCoordinatePlane::layoutDiagrams(), KDChart::AbstractCoordinatePlane::layoutPlanes(), and KDChart::AbstractCoordinatePlane::takeDiagram().

```
{
    if( diagram && oldDiagram_ != diagram ){
        AbstractDiagram* oldDiagram = oldDiagram_;
        if( d->diagrams.count() ){
            if( ! oldDiagram )
                oldDiagram = d->diagrams.first();
            takeDiagram( oldDiagram );
        }
        delete oldDiagram;
        addDiagram( diagram );
        layoutDiagrams();
        layoutPlanes(); // there might be new axes, etc
        update();
    }
}
```

7.45.4.52 void KDChart::PolarCoordinatePlane::resetGridAttributes (bool *circular*)

Reset the attributes to be used for grid lines drawn in circular direction (or in sagittal direction, resp.).

By calling this method you specify that the global attributes set by AbstractCoordinatePlane::setGlobalGridAttributes be used.

See also:

setGridAttributes, gridAttributes
AbstractCoordinatePlane::globalGridAttributes
hasOwnGridAttributes

Definition at line 380 of file KDChartPolarCoordinatePlane.cpp.

```
{
    setHasOwnGridAttributes( circular, false );
    update();
}
```

7.45.4.53 void PolarCoordinatePlane::resizeEvent (QResizeEvent ∗) `[protected]`

Definition at line 232 of file KDChartPolarCoordinatePlane.cpp.

References d, and layoutDiagrams().

```
{
    d->initialResizeEventReceived = true;
    layoutDiagrams();
}
```

7.45.4.54 int AbstractArea::rightOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the right edge of the area.

Note:

> The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 85 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
    // Re-calculate the sizes,
    // so we also get the amountOf..Overlap members set newly:
    if( ! doNotRecalculate )
        sizeHint();
    return d->amountOfRightOverlap;
}
```

7.45.4.55 void AbstractAreaBase::setBackgroundAttributes (const BackgroundAttributes & *a*) `[inherited]`

Definition at line 107 of file KDChartAbstractAreaBase.cpp.

References d.

```
{
    d->backgroundAttributes = a;
}
```

7.45.4.56 void AbstractAreaBase::setFrameAttributes (const FrameAttributes & *a*) `[inherited]`

Definition at line 97 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone().

```
{
    d->frameAttributes = a;
}
```

7.45.4.57 void KDChart::AbstractCoordinatePlane::setGeometry (const QRect & *r*) `[virtual, inherited]`

pure virtual in QLayoutItem

Note:

Do not call this function directly, unless you know exactly what you are doing. Geometry management is done by KD Chart's internal layouting measures.

Definition at line 232 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
{
//    qDebug() << "KDChart::AbstractCoordinatePlane::setGeometry(" << r << ") called";
    if( d->geometry != r ){
        d->geometry = r;
        // Note: We do *not* call update() here
        //       because it would invoke KDChart::update() recursively.
    }
}
```

7.45.4.58 void KDChart::AbstractCoordinatePlane::setGlobalGridAttributes (const GridAttributes &) `[inherited]`

Set the grid attributes to be used by this coordinate plane.

To disable grid painting, for example, your code should like this:

```
GridAttributes ga = plane->globalGridAttributes();
ga.setGlobalGridVisible( false );
plane->setGlobalGridAttributes( ga );
```

See also:

globalGridAttributes
CartesianCoordinatePlane::setGridAttributes

Definition at line 151 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
{
    d->gridAttributes = a;
    update();
}
```

7.45.4.59 void KDChart::PolarCoordinatePlane::setGridAttributes (bool *circular*, const GridAttributes &)

Set the attributes to be used for grid lines drawn in circular direction (or in sagittal direction, resp.).

To disable circular grid painting, for example, your code should like this:

```
GridAttributes ga = plane->gridAttributes( bool );
ga.setGridVisible( false );
plane-setGridAttributes( bool, ga );
```

Note:

setGridAttributes overwrites the global attributes that were set by AbstractCoordinatePlane::setGlobalGridAttributes. To re-activate these global attributes you can call resetGridAttributes.

See also:

resetGridAttributes, gridAttributes
AbstractCoordinatePlane::setGlobalGridAttributes
hasOwnGridAttributes

Definition at line 367 of file KDChartPolarCoordinatePlane.cpp.

References d, and KDChart::AbstractCoordinatePlane::propertiesChanged().

```
{
    if( circular )
        d->gridAttributesCircular = a;
    else
        d->gridAttributesSagittal = a;
    setHasOwnGridAttributes( circular, true );
    update();
    emit propertiesChanged();
}
```

7.45.4.60 void KDChart::AbstractCoordinatePlane::setGridNeedsRecalculate () `[inherited]`

Used by the chart to clear the cached grid data.

Definition at line 170 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::Chart::resizeEvent().

```
{
    d->grid->setNeedRecalculate();
}
```

7.45.4.61 void KDChart::AbstractCoordinatePlane::setParent (Chart ∗ *parent*) `[inherited]`

Called internally by KDChart::Chart.

Definition at line 185 of file KDChartAbstractCoordinatePlane.cpp.

References d.

Referenced by KDChart::Chart::addCoordinatePlane(), and KDChart::Chart::takeCoordinatePlane().

```
{
    d->parent = parent;
}
```

7.45.4.62 void KDChart::AbstractLayoutItem::setParentLayout (QLayout ∗ *lay*) `[inherited]`

Definition at line 70 of file KDChartLayoutItems.h.

```
        {
            mParentLayout = lay;
        }
```

7.45.4.63 void KDChart::AbstractLayoutItem::setParentWidget (QWidget ∗ *widget*) `[virtual, inherited]`

Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

Thus, you need to call setParentWidget on every item, that has a non-fixed size.

Definition at line 64 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::mParent.

Referenced by KDChart::Legend::buildLegend(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
{
    mParent = widget;
}
```

7.45.4.64 void KDChart::AbstractCoordinatePlane::setReferenceCoordinatePlane (AbstractCoordinatePlane ∗ *plane*) `[inherited]`

Set another coordinate plane to be used as the reference plane for this one.

Parameters:

plane The coordinate plane to be used the reference plane for this one.

See also:

referenceCoordinatePlane

Definition at line 175 of file KDChartAbstractCoordinatePlane.cpp.

References d.

```
{
    d->referenceCoordinatePlane = plane;
}
```

7.45.4.65 void PolarCoordinatePlane::setStartPosition (qreal *degrees*)

Specify the rotation of the coordinate plane.

In a Pie diagram this indicates the position where the first pie starts, in a Polar diagram it specifies the Zero position of the circular axis.

See also:
startPosition

Definition at line 313 of file KDChartPolarCoordinatePlane.cpp.

References d, and KDChart::AbstractCoordinatePlane::diagram().

```
314 {
315     Q_ASSERT_X ( diagram(), "PolarCoordinatePlane::setStartPosition",
316                  "setStartPosition() needs a diagram to be associated to the plane." );
317     d->coordinateTransformations[0].startPosition = degrees;
318 }
```

7.45.4.66 void PolarCoordinatePlane::setZoomCenter (QPointF *center*) `[virtual]`

Set the point (in value coordinates) to be used as the center point in zoom operations.

Parameters:
center The point to use.

Reimplemented from KDChart::AbstractCoordinatePlane.

Definition at line 352 of file KDChartPolarCoordinatePlane.cpp.

References d.

```
353 {
354     d->coordinateTransformations[0].zoom.xCenter = center.x();
355     d->coordinateTransformations[0].zoom.yCenter = center.y();
356 }
```

7.45.4.67 void PolarCoordinatePlane::setZoomFactorX (double *factor*) `[virtual]`

Sets the zoom factor in horizontal direction, that is applied to all coordinate transformations.

Reimplemented from KDChart::AbstractCoordinatePlane.

Definition at line 337 of file KDChartPolarCoordinatePlane.cpp.

References d.

```
338 {
339     d->coordinateTransformations[0].zoom.xFactor = factor;
340 }
```

7.45.4.68 void PolarCoordinatePlane::setZoomFactorY (double *factor*) `[virtual]`

Sets the zoom factor in vertical direction, that is applied to all coordinate transformations.

Reimplemented from KDChart::AbstractCoordinatePlane.

Definition at line 342 of file KDChartPolarCoordinatePlane.cpp.

References d.

```
{
    d->coordinateTransformations[0].zoom.yFactor = factor;
}
```

7.45.4.69 QSize KDChart::AbstractCoordinatePlane::sizeHint () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 225 of file KDChartAbstractCoordinatePlane.cpp.

References KDChart::AbstractCoordinatePlane::maximumSize().

```
{
    // we return our maxiumu (which is the full size of the Chart)
    // even if we know the plane will be smaller
    return maximumSize();
}
```

7.45.4.70 void KDChart::AbstractLayoutItem::sizeHintChanged () const `[virtual, inherited]`

Report changed size hint: ask the parent widget to recalculate the layout.

Definition at line 86 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextLayoutItem::sizeHint().

```
{
    // This is exactly like what QWidget::updateGeometry does.
//  qDebug("KDChart::AbstractLayoutItem::sizeHintChanged() called");
    if( mParent ) {
        if ( mParent->layout() )
            mParent->layout()->invalidate();
        else
            QApplication::postEvent( mParent, new QEvent( QEvent::LayoutRequest ) );
    }
}
```

7.45.4.71 QSizePolicy KDChart::AbstractCoordinatePlane::sizePolicy () const `[virtual, inherited]`

[reimplemented]

Definition at line 146 of file KDChartAbstractCoordinatePlane.cpp.

```
{
    return QSizePolicy( QSizePolicy::MinimumExpanding, QSizePolicy::MinimumExpanding );
}
```

7.45.4.72 qreal PolarCoordinatePlane::startPosition () const

Retrieve the rotation of the coordinate plane.

See also:

setStartPosition

Definition at line 320 of file KDChartPolarCoordinatePlane.cpp.

References d.

Referenced by layoutDiagrams(), and KDChart::PieDiagram::paint().

```
{
    return d->coordinateTransformations.size()
        ?  d->coordinateTransformations[0].startPosition
        :  0.0;
}
```

7.45.4.73 void AbstractCoordinatePlane::takeDiagram (AbstractDiagram ∗ *diagram*) `[virtual, inherited]`

Removes the diagram from the plane, without deleting it.

The plane no longer owns the diagram, so it is the caller's responsibility to delete the diagram.

See also:

addDiagram, replaceDiagram

Definition at line 100 of file KDChartAbstractCoordinatePlane.cpp.

References d, KDChart::AbstractCoordinatePlane::layoutDiagrams(), and KDChart::AbstractDiagram::setCoordinatePlane().

Referenced by KDChart::AbstractCoordinatePlane::replaceDiagram().

```
{
    const int idx = d->diagrams.indexOf( diagram );
    if( idx != -1 ){
        d->diagrams.removeAt( idx );
        diagram->setParent( 0 );
        diagram->setCoordinatePlane( 0 );
        layoutDiagrams();
        update();
    }
}
```

7.45.4.74 int AbstractArea::topOverlap (bool *doNotRecalculate* = false) const `[virtual, inherited]`

This is called at layout time by KDChart::AutoSpacerLayoutItem::sizeHint().

The method triggers AbstractArea::sizeHint() to find out the amount of overlap at the top edge of the area.

Note:

The default implementation is not using any caching, it might make sense to implement a more sophisticated solution for derived classes that have complex work to do in sizeHint(). All we have here is a primitive flag to be set by the caller if it is sure that no sizeHint() needs to be called.

Definition at line 93 of file KDChartAbstractArea.cpp.

References d.

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
    // Re-calculate the sizes,
    // so we also get the amountOf..Overlap members set newly:
    if( ! doNotRecalculate )
        sizeHint();
    return d->amountOfTopOverlap;
}
```

7.45.4.75 const QPointF PolarCoordinatePlane::translate (const QPointF & *diagramPoint*) const `[virtual]`

Translate the given point in value space coordinates to a position in pixel space.

Parameters:

diagramPoint The point in value coordinates.

Returns:

The translated point.

Implements KDChart::AbstractCoordinatePlane.

Definition at line 280 of file KDChartPolarCoordinatePlane.cpp.

References d.

Referenced by buildReferenceRect().

```
{
    Q_ASSERT_X ( d->currentTransformation != 0, "PolarCoordinatePlane::translate",
                 "Only call translate() from within paint()." );
    return  d->currentTransformation->translate ( diagramPoint );
}
```

7.45.4.76 const QPointF PolarCoordinatePlane::translatePolar (const QPointF & *diagramPoint*) const

Definition at line 287 of file KDChartPolarCoordinatePlane.cpp.

References d.

```
{
    Q_ASSERT_X ( d->currentTransformation != 0, "PolarCoordinatePlane::translate",
                 "Only call translate() from within paint()." );
    return  d->currentTransformation->translatePolar ( diagramPoint );
}
```

7.45.4.77 QPointF PolarCoordinatePlane::zoomCenter () const `[virtual]`

Returns:

The center point (in value coordinates) of the coordinate plane, that is used for zoom operations.

Reimplemented from KDChart::AbstractCoordinatePlane.

Definition at line 347 of file KDChartPolarCoordinatePlane.cpp.

References d.

```
{
    return QPointF( d->coordinateTransformations[0].zoom.xCenter, d->coordinateTransformations[0].zoom
}
```

7.45.4.78 double PolarCoordinatePlane::zoomFactorX () const `[virtual]`

Returns:

The zoom factor in horizontal direction, that is applied to all coordinate transformations.

Reimplemented from KDChart::AbstractCoordinatePlane.

Definition at line 327 of file KDChartPolarCoordinatePlane.cpp.

References d.

```
{
    return d->coordinateTransformations[0].zoom.xFactor;
}
```

7.45.4.79 double PolarCoordinatePlane::zoomFactorY () const `[virtual]`

Returns:

The zoom factor in vertical direction, that is applied to all coordinate transformations.

Reimplemented from KDChart::AbstractCoordinatePlane.

Definition at line 332 of file KDChartPolarCoordinatePlane.cpp.

References d.

```
{
    return d->coordinateTransformations[0].zoom.yFactor;
}
```

7.45.5 Member Data Documentation

7.45.5.1 Q_SIGNALS KDChart::AbstractCoordinatePlane::__pad0__ `[inherited]`

Reimplemented from KDChart::AbstractArea.

Definition at line 297 of file KDChartAbstractCoordinatePlane.h.

7.45.5.2 QWidget∗ KDChart::AbstractLayoutItem::mParent `[protected, inherited]`

Definition at line 88 of file KDChartLayoutItems.h.

Referenced by KDChart::AbstractLayoutItem::setParentWidget().

7.45.5.3 QLayout∗ KDChart::AbstractLayoutItem::mParentLayout `[protected, inherited]`

Definition at line 89 of file KDChartLayoutItems.h.

7.45.5.4 protected KDChart::PolarCoordinatePlane::Q_SLOTS `[protected]`

Reimplemented from KDChart::AbstractCoordinatePlane.

Definition at line 154 of file KDChartPolarCoordinatePlane.h.

The documentation for this class was generated from the following files:

- KDChartPolarCoordinatePlane.h
- KDChartPolarCoordinatePlane.cpp

7.46 KDChart::PolarDiagram Class Reference

`#include <KDChartPolarDiagram.h>`

Inheritance diagram for KDChart::PolarDiagram:Collaboration diagram for KDChart::PolarDiagram:

Public Member Functions

- bool allowOverlappingDataValueTexts () const
- bool antiAliasing () const
- virtual AttributesModel ∗ attributesModel () const

  Returns the AttributesModel, that is used by this diagram.

- QBrush brush (const QModelIndex &index) const

  Retrieve the brush to be used, for painting the datapoint at the given index in the model.

- QBrush brush (int dataset) const

  Retrieve the brush to be used for the given dataset.

- QBrush brush () const

  Retrieve the brush to be used for painting datapoints globally.

- virtual PolarDiagram ∗ clone () const
- bool closeDatasets () const
- int columnCount () const
- bool compare (const AbstractDiagram ∗other) const

  Returns true if both diagrams have the same settings.

- AbstractCoordinatePlane ∗ coordinatePlane () const

  The coordinate plane associated with the diagram.

- const QPair< QPointF, QPointF > dataBoundaries () const

  Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

- virtual void dataChanged (const QModelIndex &topLeft, const QModelIndex &bottomRight)

  [reimplemented]

- QList< QBrush > datasetBrushes () const

  The set of dataset brushes currently used, for use in legends, etc.

- int datasetDimension () const

  The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

- QStringList datasetLabels () const

  The set of dataset labels currently displayed, for use in legends, etc.

- QList< MarkerAttributes > datasetMarkers () const

  The set of dataset markers currently used, for use in legends, etc.

- QList< QPen > datasetPens () const

  The set of dataset pens currently used, for use in legends, etc.

- DataValueAttributes dataValueAttributes (const QModelIndex &index) const

  Retrieve the DataValueAttributes for the given index.

- DataValueAttributes dataValueAttributes (int column) const

  Retrieve the DataValueAttributes for the given dataset.

- DataValueAttributes dataValueAttributes () const

  Retrieve the DataValueAttributes speficied globally.

- virtual void doItemsLayout ()

  [reimplemented]

- virtual int horizontalOffset () const

  [reimplemented]

- virtual QModelIndex indexAt (const QPoint &point) const

  [reimplemented]

- bool isHidden (const QModelIndex &index) const

  Retrieve the hidden status for the given index.

- bool isHidden (int column) const

  Retrieve the hidden status for the given dataset.

- bool isHidden () const

  Retrieve the hidden status speficied globally.

- virtual bool isIndexHidden (const QModelIndex &index) const

  [reimplemented]

- QStringList itemRowLabels () const

  The set of item row labels currently displayed, for use in Abscissa axes, etc.

- virtual QModelIndex moveCursor (CursorAction cursorAction, Qt::KeyboardModifiers modifiers)

  [reimplemented]

- virtual double numberOfGridRings () const

  [reimplemented]

- virtual double numberOfValuesPerDataset () const

  [reimplemented]

- void paintDataValueText (QPainter ∗painter, const QModelIndex &index, const QPointF &pos, double value)
- void paintMarker (QPainter ∗painter, const QModelIndex &index, const QPointF &pos)

- virtual void paintMarker (QPainter ∗painter, const MarkerAttributes &markerAttributes, const QBrush &brush, const QPen &, const QPointF &point, const QSizeF &size)
- QPen pen (const QModelIndex &index) const

  Retrieve the pen to be used, for painting the datapoint at the given index in the model.

- QPen pen (int dataset) const

  Retrieve the pen to be used for the given dataset.

- QPen pen () const

  Retrieve the pen to be used for painting datapoints globally.

- bool percentMode () const
- const PolarCoordinatePlane ∗ polarCoordinatePlane () const
- PolarDiagram (QWidget ∗parent=0, PolarCoordinatePlane ∗plane=0)
- virtual void resize (const QSizeF &area)

  [reimplemented]

- bool rotateCircularLabels () const
- virtual void scrollTo (const QModelIndex &index, ScrollHint hint=EnsureVisible)

  [reimplemented]

- void setAllowOverlappingDataValueTexts (bool allow)

  Set whether data value labels are allowed to overlap.

- void setAntiAliasing (bool enabled)

  Set whether anti-aliasing is to be used while rendering this diagram.

- virtual void setAttributesModel (AttributesModel ∗model)

  Associate an AttributesModel with this diagram.

- void setBrush (const QBrush &brush)

  Set the brush to be used, for painting all datasets in the model.

- void setBrush (int dataset, const QBrush &brush)

  Set the brush to be used, for painting the given dataset.

- void setBrush (const QModelIndex &index, const QBrush &brush)

  Set the brush to be used, for painting the datapoint at the given index.

- void setCloseDatasets (bool closeDatasets)

  Close each of the data series by connecting the last point to its respective start point.

- virtual void setCoordinatePlane (AbstractCoordinatePlane ∗plane)

  Set the coordinate plane associated with the diagram.

- void setDatasetDimension (int dimension)

  Sets the dataset dimension of the diagram.

- void setDataValueAttributes (const DataValueAttributes &a)

  Set the DataValueAttributes for all datapoints in the model.

- void setDataValueAttributes (int dataset, const DataValueAttributes &a)

  Set the DataValueAttributes for the given dataset.

- void setDataValueAttributes (const QModelIndex &index, const DataValueAttributes &a)

  Set the DataValueAttributes for the given index.

- void setHidden (bool hidden)

  Hide (or unhide, resp.) all datapoints in the model.

- void setHidden (int column, bool hidden)

  Hide (or unhide, resp.) a dataset.

- void setHidden (const QModelIndex &index, bool hidden)

  Hide (or unhide, resp.) a data cell.

- virtual void setModel (QAbstractItemModel ∗model)

  Associate a model with the diagram.

- void setPen (const QPen &pen)

  Set the pen to be used, for painting all datasets in the model.

- void setPen (int dataset, const QPen &pen)

  Set the pen to be used, for painting the given dataset.

- void setPen (const QModelIndex &index, const QPen &pen)

  Set the pen to be used, for painting the datapoint at the given index.

- void setPercentMode (bool percent)
- virtual void setRootIndex (const QModelIndex &idx)

  Set the root index in the model, where the diagram starts referencing data for display.

- void setRotateCircularLabels (bool rotateCircularLabels)
- virtual void setSelection (const QRect &rect, QItemSelectionModel::SelectionFlags command)

  [reimplemented]

- void setShowDelimitersAtPosition (Position position, bool showDelimiters)
- void setShowLabelsAtPosition (Position position, bool showLabels)
- void setZeroDegreePosition (int degrees)
- bool showDelimitersAtPosition (Position position) const
- bool showLabelsAtPosition (Position position) const
- void update () const
- void useDefaultColors ()

  Set the palette to be used, for painting datasets to the default palette.

- void useRainbowColors ()

  Set the palette to be used, for painting datasets to the rainbow palette.

- virtual bool usesExternalAttributesModel () const

Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

- void useSubduedColors ()

  Set the palette to be used, for painting datasets to the subdued palette.

- virtual double valueTotals () const

  [reimplemented]

- virtual int verticalOffset () const

  [reimplemented]

- virtual QRect visualRect (const QModelIndex &index) const

  [reimplemented]

- virtual QRegion visualRegionForSelection (const QItemSelection &selection) const

  [reimplemented]

- int zeroDegreePosition () const
- virtual ∼PolarDiagram ()

Protected Member Functions

- QModelIndex attributesModelRootIndex () const
- virtual const QPair< QPointF, QPointF > calculateDataBoundaries () const

  [reimplemented]

- virtual bool checkInvariants (bool justReturnTheStatus=false) const
- QModelIndex columnToIndex (int column) const
- void dataHidden ()

  This signal is emitted, when the hidden status of at least one data cell was (un)set.

- void modelsChanged ()

  This signal is emitted, when either the model or the AttributesModel is replaced.

- virtual void paint (PaintContext ∗paintContext)

  [reimplemented]

- virtual void paintDataValueTexts (QPainter ∗painter)
- void paintEvent (QPaintEvent ∗)
- virtual void paintMarkers (QPainter ∗painter)
- virtual void paintPolarMarkers (PaintContext ∗ctx, const QPolygonF &polygon)
- void propertiesChanged ()

  Emitted upon change of a property of the Diagram.

- void resizeEvent (QResizeEvent ∗)
- void setAttributesModelRootIndex (const QModelIndex &)
- void setDataBoundariesDirty () const
- double valueForCell (int row, int column) const

  Helper method, retrieving the data value (DisplayRole) for a given row and column.

Protected Attributes

- Q_SIGNALS __pad0__: void layoutChanged(AbstractDiagram∗)

7.46.1 Constructor & Destructor Documentation

7.46.1.1 PolarDiagram::PolarDiagram (QWidget ∗ *parent* = 0, PolarCoordinatePlane ∗ *plane* = 0) `[explicit]`

Definition at line 49 of file KDChartPolarDiagram.cpp.

Referenced by clone().

```
49                                                                                            :
50      AbstractPolarDiagram( new Private( ), parent, plane )
51 {
52 }
```

7.46.1.2 PolarDiagram::∼PolarDiagram () `[virtual]`

Definition at line 54 of file KDChartPolarDiagram.cpp.

```
55 {
56 }
```

7.46.2 Member Function Documentation

7.46.2.1 bool AbstractDiagram::allowOverlappingDataValueTexts () const `[inherited]`

Returns:

Whether data value labels are allowed to overlap.

Definition at line 446 of file KDChartAbstractDiagram.cpp.

References d.

```
450 {
```

7.46.2.2 bool AbstractDiagram::antiAliasing () const `[inherited]`

Returns:

Whether anti-aliasing is to be used for rendering this diagram.

Definition at line 457 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::paint().

```
461 {
```

7.46.2.3 AttributesModel ∗ AbstractDiagram::attributesModel () const `[virtual, inherited]`

Returns the AttributesModel, that is used by this diagram.

By default each diagram owns its own AttributesModel, which should never be deleted. Only if a user-supplied AttributesModel has been set does the pointer returned here not belong to the diagram.

Returns:
The AttributesModel associated with the diagram.

See also:
setAttributesModel

Definition at line 286 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::RingDiagram::paint(), paint(), and KDChart::BarDiagram::setBarAttributes().

```
287 {
288     return d->attributesModel;
289 }
```

7.46.2.4 QModelIndex AbstractDiagram::attributesModelRootIndex () const `[protected, inherited]`

returns a QModelIndex pointing into the AttributesModel that corresponds to the root index of the diagram.

Definition at line 310 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::calculateDataBoundaries(), KDChart::BarDiagram::calculateDataBoundaries(), KDChart::LineDiagram::numberOfAbscissaSegments(), KDChart::BarDiagram::numberOfAbscissaSegments(), KDChart::LineDiagram::numberOfOrdinateSegments(), KDChart::BarDiagram::numberOfOrdinateSegments(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and KDChart::AbstractDiagram::valueForCell().

```
316 {
```

7.46.2.5 QBrush AbstractDiagram::brush (const QModelIndex & *index*) const `[inherited]`

Retrieve the brush to be used, for painting the datapoint at the given index in the model.

Parameters:
index The index of the datapoint in the model.

Returns:
The brush to use for painting.

Definition at line 816 of file KDChartAbstractDiagram.cpp.

```
822                                :
QRect AbstractDiagram::visualRect(const QModelIndex &) const
```

7.46.2.6 QBrush AbstractDiagram::brush (int *dataset*) const `[inherited]`

Retrieve the brush to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the brush for.

Returns:

The brush to use for painting.

Definition at line 808 of file KDChartAbstractDiagram.cpp.

```
815 {
```

7.46.2.7 QBrush AbstractDiagram::brush () const `[inherited]`

Retrieve the brush to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The brush to use for painting.

Definition at line 802 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), and KDChart::AbstractDiagram::paintMarker().

```
807 {
```

7.46.2.8 const QPair< QPointF, QPointF > PolarDiagram::calculateDataBoundaries () const `[protected, virtual]`

[reimplemented]

Implements KDChart::AbstractDiagram.

Definition at line 100 of file KDChartPolarDiagram.cpp.

References KDChart::AbstractDiagram::checkInvariants().

```
101 {
102     if ( !checkInvariants(true) ) return QPair<QPointF, QPointF>( QPointF( 0, 0 ), QPointF( 0, 0 ) );
103     const int rowCount = model()->rowCount(rootIndex());
104     const int colCount = model()->columnCount(rootIndex());
105     double xMin = 0.0;
106     double xMax = colCount;
107     double yMin = 0, yMax = 0;
108     for ( int j=0; j<colCount; ++j ) {
109         for ( int i=0; i< rowCount; ++i ) {
110             double value = model()->data( model()->index( i, j, rootIndex() ) ).toDouble();
111             yMax = qMax( yMax, value );
112         }
113     }
114     QPointF bottomLeft ( QPointF( xMin, yMin ) );
115     QPointF topRight ( QPointF( xMax, yMax ) );
116     return QPair<QPointF, QPointF> ( bottomLeft,  topRight );
117 }
```

7.46.2.9 bool AbstractDiagram::checkInvariants (bool *justReturnTheStatus* = false) const `[protected, virtual, inherited]`

Definition at line 930 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane().

Referenced by KDChart::RingDiagram::calculateDataBoundaries(), calculateDataBoundaries(), KDChart::PieDiagram::calculateDataBoundaries(), KDChart::LineDiagram::calculateDataBoundaries(), KDChart::BarDiagram::calculateDataBoundaries(), KDChart::RingDiagram::paint(), paint(), KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and KDChart::AbstractDiagram::paintMarker().

```
930                                 {
931         Q_ASSERT_X ( model(), "AbstractDiagram::checkInvariants()",
932                     "There is no usable model set, for the diagram." );
933
934         Q_ASSERT_X ( coordinatePlane(), "AbstractDiagram::checkInvariants()",
935                     "There is no usable coordinate plane set, for the diagram." );
936     }
937     return model() && coordinatePlane();
938 }
939
940 int AbstractDiagram::datasetDimension( ) const
```

7.46.2.10 **PolarDiagram** ∗ PolarDiagram::clone () const `[virtual]`

Definition at line 89 of file KDChartPolarDiagram.cpp.

References closeDatasets(), d, PolarDiagram(), rotateCircularLabels(), showDelimitersAtPosition(), and showLabelsAtPosition().

```
90 {
91     PolarDiagram* newDiagram = new PolarDiagram( new Private( *d ) );
92     // This needs to be copied after the fact
93     newDiagram->d->showDelimitersAtPosition = d->showDelimitersAtPosition;
94     newDiagram->d->showLabelsAtPosition = d->showLabelsAtPosition;
95     newDiagram->d->rotateCircularLabels = d->rotateCircularLabels;
96     newDiagram->d->closeDatasets = d->closeDatasets;
97     return newDiagram;
98 }
```

7.46.2.11 bool PolarDiagram::closeDatasets () const

Definition at line 239 of file KDChartPolarDiagram.cpp.

References d.

Referenced by clone(), and paint().

```
240 {
241     return d->closeDatasets;
242 }
```

7.46.2.12 int AbstractPolarDiagram::columnCount () const `[inherited]`

Definition at line 60 of file KDChartAbstractPolarDiagram.cpp.

References KDChart::AbstractPolarDiagram::numberOfValuesPerDataset().

Referenced by KDChart::PieDiagram::calculateDataBoundaries(), KDChart::PieDiagram::paint(), and KDChart::PieDiagram::valueTotals().

```
61 {
62     return static_cast<int>( numberOfValuesPerDataset() );
63 }
```

7.46.2.13 QModelIndex AbstractDiagram::columnToIndex (int *column*) const `[protected, inherited]`

Definition at line 317 of file KDChartAbstractDiagram.cpp.

```
323 {
```

7.46.2.14 bool AbstractDiagram::compare (const AbstractDiagram ∗ *other*) const `[inherited]`

Returns true if both diagrams have the same settings.

Definition at line 135 of file KDChartAbstractDiagram.cpp.

```
136 {
137     if( other == this ) return true;
138     if( ! other ){
139         //qDebug() << "AbstractDiagram::compare() cannot compare to Null pointer";
140         return false;
141     }
142     /*
143     qDebug() << "\n             AbstractDiagram::compare() QAbstractScrollArea:";
144             // compare QAbstractScrollArea properties
145     qDebug() <<
146             ((horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
147             (verticalScrollBarPolicy()    == other->verticalScrollBarPolicy()));
148     qDebug() << "AbstractDiagram::compare() QFrame:";
149             // compare QFrame properties
150     qDebug() <<
151             ((frameShadow() == other->frameShadow()) &&
152             (frameShape()   == other->frameShape()) &&
153             (frameWidth()   == other->frameWidth()) &&
154             (lineWidth()    == other->lineWidth()) &&
155             (midLineWidth() == other->midLineWidth()));
156     qDebug() << "AbstractDiagram::compare() QAbstractItemView:";
157             // compare QAbstractItemView properties
158     qDebug() <<
159             ((alternatingRowColors() == other->alternatingRowColors()) &&
160             (hasAutoScroll()         == other->hasAutoScroll()) &&
161 #if QT_VERSION > 0x040199
162             (dragDropMode()          == other->dragDropMode()) &&
163             (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
164             (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
165             (verticalScrollMode()    == other->verticalScrollMode()) &&
166 #endif
167             (dragEnabled()           == other->dragEnabled()) &&
            (editTriggers()          == other->editTriggers()) &&
            (iconSize()              == other->iconSize()) &&
            (selectionBehavior()     == other->selectionBehavior()) &&
            (selectionMode()         == other->selectionMode()) &&
            (showDropIndicator()     == other->showDropIndicator()) &&
            (tabKeyNavigation()      == other->tabKeyNavigation()) &&
            (textElideMode()         == other->textElideMode()));
    qDebug() << "AbstractDiagram::compare() AttributesModel: ";
            // compare all of the properties stored in the attributes model
    qDebug() << attributesModel()->compare( other->attributesModel() );
    qDebug() << "AbstractDiagram::compare() own:";
            // compare own properties
    qDebug() <<
            ((rootIndex().column()            == other->rootIndex().column()) &&
            (rootIndex().row()                == other->rootIndex().row()) &&
            (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
            (antiAliasing()                   == other->antiAliasing()) &&
            (percentMode()                    == other->percentMode()) &&
            (datasetDimension()               == other->datasetDimension()));
    */
    return  // compare QAbstractScrollArea properties
            (horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
            (verticalScrollBarPolicy()   == other->verticalScrollBarPolicy()) &&
            // compare QFrame properties
            (frameShadow()  == other->frameShadow()) &&
            (frameShape()   == other->frameShape()) &&
            (frameWidth()   == other->frameWidth()) &&
            (lineWidth()    == other->lineWidth()) &&
            (midLineWidth() == other->midLineWidth()) &&
            // compare QAbstractItemView properties
            (alternatingRowColors()  == other->alternatingRowColors()) &&
            (hasAutoScroll()         == other->hasAutoScroll()) &&
#if QT_VERSION > 0x040199
            (dragDropMode()          == other->dragDropMode()) &&
            (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
            (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
            (verticalScrollMode()    == other->verticalScrollMode()) &&
#endif
            (dragEnabled()           == other->dragEnabled()) &&
            (editTriggers()          == other->editTriggers()) &&
            (iconSize()              == other->iconSize()) &&
            (selectionBehavior()     == other->selectionBehavior()) &&
            (selectionMode()         == other->selectionMode()) &&
            (showDropIndicator()     == other->showDropIndicator()) &&
            (tabKeyNavigation()      == other->tabKeyNavigation()) &&
            (textElideMode()         == other->textElideMode()) &&
            // compare all of the properties stored in the attributes model
            attributesModel()->compare( other->attributesModel() ) &&
            // compare own properties
            (rootIndex().column()             == other->rootIndex().column()) &&
            (rootIndex().row()                == other->rootIndex().row()) &&
            (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
            (antiAliasing()                   == other->antiAliasing()) &&
            (percentMode()                    == other->percentMode()) &&
            (datasetDimension()               == other->datasetDimension());
}
```

7.46.2.15 AbstractCoordinatePlane ∗ AbstractDiagram::coordinatePlane () const `[inherited]`

The coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. By default this is a CartesianCoordinatePlane.

Returns:

The coordinate plane associated with the diagram.

Definition at line 226 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractCartesianDiagram::layoutPlanes(), paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), KDChart::AbstractPolarDiagram::polarCoordinatePlane(), and KDChart::AbstractCartesianDiagram::setCoordinatePlane().

```
227 {
228     return d->plane;
229 }
```

7.46.2.16 const QPair< QPointF, QPointF > AbstractDiagram::dataBoundaries () const `[inherited]`

Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

This method returns a chached result of calculations done by calculateDataBoundaries. Classes derived from AbstractDiagram must implement the calculateDataBoundaries function, to specify their own way of calculating the data boundaries. If derived classes want to force recalculation of the data boundaries, they can call setDataBoundariesDirty()

Returned value is in diagram coordinates.

Definition at line 231 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::calculateDataBoundaries(), and d.

Referenced by KDChart::CartesianCoordinatePlane::getRawDataBoundingRectFromDiagrams(), KDChart::PolarCoordinatePlane::layoutDiagrams(), KDChart::LineDiagram::paint(), and KDChart::BarDiagram::paint().

```
232 {
233     if( d->databoundariesDirty ){
234         d->databoundaries = calculateDataBoundaries ();
235         d->databoundariesDirty = false;
236     }
237     return d->databoundaries;
238 }
```

7.46.2.17 void AbstractDiagram::dataChanged (const QModelIndex & *topLeft*, const QModelIndex & *bottomRight*) `[virtual, inherited]`

[reimplemented]

Definition at line 338 of file KDChartAbstractDiagram.cpp.

References d.

```
338 {
339   // We are still too dumb to do intelligent updates...
340   d->databoundariesDirty = true;
341   scheduleDelayedItemsLayout();
342 }
343
344
```

7.46.2.18 void KDChart::AbstractDiagram::dataHidden () `[protected, inherited]`

This signal is emitted, when the hidden status of at least one data cell was (un)set.

7.46.2.19 QList< QBrush > AbstractDiagram::datasetBrushes () const `[inherited]`

The set of dataset brushes currently used, for use in legends, etc.

Note:

Cell-level override brushes, if set, take precedence over the dataset values, so you might need to check these too, in order to find the brush, that is used for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 894 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), KDChart::Legend::datasetCount(), and KDChart::Legend::setBrushesFromDiagram().

```
896                                                                                   {
897         QBrush brush = qVariantValue<QBrush>( attributesModel()->headerData( i, Qt::Vertical, Dataset
898         ret << brush;
899     }
900
901     return ret;
902 }
903
904 QList<QPen> AbstractDiagram::datasetPens() const
```

7.46.2.20 int AbstractDiagram::datasetDimension () const `[inherited]`

The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

For each dimension it will expect one column of values in the model. If the dimensionality is 1, automatic values will be used for the abscissa.

For example a diagram with the default dimension of 1, will have one column per datapoint (the y values) and will use automatic values for the x axis (1, 2, 3, ... n). If the dimension is 2, the diagram will use the first, (and the third, fifth, etc) columns as X values, and the second, (and the fourth, sixth, etc) column as Y values.

Returns:

The dataset dimension of the diagram.

Definition at line 942 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::calculateDataBoundaries(), KDChart::LineDiagram::getCellValues(), KDChart::CartesianCoordinatePlane::getDataDimensionsList(), KDChart::LineDiagram::paint(), and KDChart::LineDiagram::setType().

```
946 {
```

7.46.2.21 QStringList AbstractDiagram::datasetLabels () const `[inherited]`

The set of dataset labels currently displayed, for use in legends, etc.

Returns:

The set of dataset labels currently displayed.

Definition at line 882 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), and KDChart::Legend::datasetCount().

```
883                                                  : " << attributesModel()->columnCount(attributesModel
884     const int columnCount = attributesModel()->columnCount(attributesModelRootIndex());
885     for( int i = datasetDimension()-1; i < columnCount; i += datasetDimension() ){
886         //qDebug() << "dataset label: " << attributesModel()->headerData( i, Qt::Horizontal, Qt::Displ
887         ret << attributesModel()->headerData( i, Qt::Horizontal, Qt::DisplayRole ).toString();
888     }
889     return ret;
890 }
891
892 QList<QBrush> AbstractDiagram::datasetBrushes() const
```

7.46.2.22 QList< MarkerAttributes > AbstractDiagram::datasetMarkers () const `[inherited]`

The set of dataset markers currently used, for use in legends, etc.

Note:

Cell-level override markers, if set, take precedence over the dataset values, so you might need to check these too, in order to find the marker, that is shown for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 917 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
919                                                                                   {
920         DataValueAttributes a =
921             qVariantValue<DataValueAttributes>( attributesModel()->headerData( i, Qt::Vertical, DataVa
922         const MarkerAttributes &ma = a.markerAttributes();
923         ret << ma;
924     }
925     return ret;
926 }
927
928 bool AbstractDiagram::checkInvariants( bool justReturnTheStatus ) const
```

7.46.2.23 QList< QPen > AbstractDiagram::datasetPens () const `[inherited]`

The set of dataset pens currently used, for use in legends, etc.

Note:

Cell-level override pens, if set, take precedence over the dataset values, so you might need to check these too, in order to find the pens, that is used for a single cell.

Returns:

The current set of dataset pens.

Definition at line 906 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
                                                                                  {
        QPen pen = qVariantValue<QPen>( attributesModel()->headerData( i, Qt::Vertical, DatasetPenRole
        ret << pen;
    }
    return ret;
}

QList<MarkerAttributes> AbstractDiagram::datasetMarkers() const
```

7.46.2.24 DataValueAttributes AbstractDiagram::dataValueAttributes (const QModelIndex & *index*) const `[inherited]`

Retrieve the DataValueAttributes for the given index.

This will fall back automatically to what was set at dataset or model level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the attributes for.

Returns:

The DataValueAttributes for the given index.

Definition at line 427 of file KDChartAbstractDiagram.cpp.

```
{
```

7.46.2.25 DataValueAttributes AbstractDiagram::dataValueAttributes (int *column*) const `[inherited]`

Retrieve the DataValueAttributes for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the attributes for.

Returns:

The DataValueAttributes for the given dataset.

Definition at line 420 of file KDChartAbstractDiagram.cpp.

```
426 {
```

7.46.2.26 DataValueAttributes AbstractDiagram::dataValueAttributes () const `[inherited]`

Retrieve the DataValueAttributes speficied globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The global DataValueAttributes.

Definition at line 414 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::AbstractDiagram::paintDataValueText(), and KDChart::AbstractDiagram::paintMarker().

```
419 {
```

7.46.2.27 void AbstractDiagram::doItemsLayout () `[virtual, inherited]`

[reimplemented]

Definition at line 329 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::AbstractDiagram::update().

```
329                     {
330         d->plane->layoutDiagrams();
331         update();
332     }
333     QAbstractItemView::doItemsLayout();
334 }
335
336 void AbstractDiagram::dataChanged( const QModelIndex &topLeft,
```

7.46.2.28 int AbstractDiagram::horizontalOffset () const `[virtual, inherited]`

[reimplemented]

Definition at line 839 of file KDChartAbstractDiagram.cpp.

```
841 { return 0; }
```

7.46.2.29 QModelIndex AbstractDiagram::indexAt (const QPoint & *point*) const `[virtual, inherited]`

[reimplemented]

Definition at line 833 of file KDChartAbstractDiagram.cpp.

```
835 { return QModelIndex(); }
```

7.46.2.30 bool AbstractDiagram::isHidden (const QModelIndex & *index*) const `[inherited]`

Retrieve the hidden status for the given index.

This will fall back automatically to what was set at dataset or diagram level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the hidden status for.

Returns:

The hidden status for the given index.

Definition at line 386 of file KDChartAbstractDiagram.cpp.

7.46.2.31 bool AbstractDiagram::isHidden (int *column*) const `[inherited]`

Retrieve the hidden status for the given dataset.

This will fall back automatically to what was set at diagram level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the hidden status for.

Returns:

The hidden status for the given dataset.

Definition at line 379 of file KDChartAbstractDiagram.cpp.

```
385 {
```

7.46.2.32 bool AbstractDiagram::isHidden () const `[inherited]`

Retrieve the hidden status speficied globally.

This will fall back automatically to the default settings (= not hidden), if there are no specific settings.

Returns:

The global hidden status.

Definition at line 373 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), KDChart::LineDiagram::paint(), and KDChart::LineDiagram::valueForCellTesting().

```
378 {
```

7.46.2.33 bool AbstractDiagram::isIndexHidden (const QModelIndex & *index*) const `[virtual, inherited]`

[reimplemented]

Definition at line 845 of file KDChartAbstractDiagram.cpp.

```
847 {}
```

7.46.2.34 QStringList AbstractDiagram::itemRowLabels () const `[inherited]`

The set of item row labels currently displayed, for use in Abscissa axes, etc.

Returns:
The set of item row labels currently displayed.

Definition at line 870 of file KDChartAbstractDiagram.cpp.

```
871                                                  : " << attributesModel()->rowCount(attributesModelRoo
872     const int rowCount = attributesModel()->rowCount(attributesModelRootIndex());
873     for( int i = 0; i < rowCount; ++i ){
874         //qDebug() << "item row label: " << attributesModel()->headerData( i, Qt::Vertical, Qt::Displa
875         ret << attributesModel()->headerData( i, Qt::Vertical, Qt::DisplayRole ).toString();
876     }
877     return ret;
878 }
879 
880 QStringList AbstractDiagram::datasetLabels() const
```

7.46.2.35 void KDChart::AbstractDiagram::modelsChanged () `[protected, inherited]`

This signal is emitted, when either the model or the AttributesModel is replaced.

Referenced by KDChart::AbstractDiagram::setAttributesModel(), and KDChart::AbstractDiagram::setModel().

7.46.2.36 QModelIndex AbstractDiagram::moveCursor (CursorAction *cursorAction*, Qt::KeyboardModifiers *modifiers*) `[virtual, inherited]`

[reimplemented]

Definition at line 836 of file KDChartAbstractDiagram.cpp.

```
838 { return 0; }
```

7.46.2.37 double PolarDiagram::numberOfGridRings () const `[virtual]`

[reimplemented]

Implements KDChart::AbstractPolarDiagram.

Definition at line 208 of file KDChartPolarDiagram.cpp.

```
209 {
210     return 5; // FIXME
211 }
```

7.46.2.38 double PolarDiagram::numberOfValuesPerDataset () const `[virtual]`

[reimplemented]

Implements KDChart::AbstractPolarDiagram.

Definition at line 202 of file KDChartPolarDiagram.cpp.

```
{
    return model() ? model()->rowCount(rootIndex()) : 0.0;
}
```

7.46.2.39 void PolarDiagram::paint (PaintContext ∗ *paintContext*) `[protected, virtual]`

[reimplemented]

Implements KDChart::AbstractDiagram.

Definition at line 145 of file KDChartPolarDiagram.cpp.

References KDChart::AbstractDiagram::attributesModel(), KDChart::AbstractDiagram::checkInvariants(), closeDatasets(), KDChart::AbstractDiagram::coordinatePlane(), KDChart::AttributesModel::headerData(), KDChart::AbstractDiagram::paintDataValueText(), KDChart::PaintContext::painter(), paintPolarMarkers(), KDChart::PaintContext::rectangle(), and KDChart::AbstractCoordinatePlane::translate().

Referenced by paintEvent().

```
{
    // note: Not having any data model assigned is no bug
    //       but we can not draw a diagram then either.
    if ( !checkInvariants(true) )
        return;

    const int rowCount = model()->rowCount( rootIndex() );
    const int colCount = model()->columnCount( rootIndex() );
    DataValueTextInfoList list;

    for ( int j=0; j < colCount; ++j ) {
        QBrush brush = qVariantValue<QBrush>( attributesModel()->headerData( j, Qt::Vertical, KDChart:
        QPolygonF polygon;
        QPointF point0;
        for ( int i=0; i < rowCount; ++i ) {
            QModelIndex index = model()->index( i, j, rootIndex() );
            const double value = model()->data( index ).toDouble();
            QPointF point = coordinatePlane()->translate( QPointF( value, i ) );
            polygon.append( point );
            const DataValueTextInfo info( index, point, point, value );
            list.append( info );
            if( ! i )
                point0= point;
        }
        if( closeDatasets() && rowCount )
            polygon.append( point0 );

        PainterSaver painterSaver( ctx->painter() );
        ctx->painter()->setRenderHint ( QPainter::Antialiasing );
        ctx->painter()->setBrush( brush );
        QPen p( ctx->painter()->pen() );
        p.setColor( brush.color() ); // FIXME use DatasetPenRole
        p.setWidth( 2 );// FIXME properties
        ctx->painter()->setPen( p );
        polygon.translate( ctx->rectangle().topLeft() );
        ctx->painter()->drawPolyline( polygon );
        paintPolarMarkers( ctx, polygon );
    }
    DataValueTextInfoListIterator it( list );
    while ( it.hasNext() ) {
        const DataValueTextInfo& info = it.next();
        paintDataValueText( ctx->painter(), info.index, info.pos, info.value );
    }
}
```

7.46.2.40 void AbstractDiagram::paintDataValueText (QPainter ∗ *painter*, const QModelIndex & *index*, const QPointF & *pos*, double *value*) `[inherited]`

Definition at line 474 of file KDChartAbstractDiagram.cpp.

References KDChart::RelativePosition::alignment(), KDChart::TextAttributes::calculatedFont(), d, KDChart::DataValueAttributes::dataLabel(), KDChart::AbstractDiagram::dataValueAttributes(), KDChart::DataValueAttributes::decimalDigits(), KDChart::TextAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::TextAttributes::pen(), KDChart::DataValueAttributes::position(), KDChart::DataValueAttributes::prefix(), KDChart::TextAttributes::rotation(), KDChart::DataValueAttributes::showRepetitiveDataLabels(), KDChart::DataValueAttributes::suffix(), and KDChart::DataValueAttributes::textAttributes().

Referenced by KDChart::RingDiagram::paint(), and paint().

```
{
    // paint one data series
    const DataValueAttributes a( dataValueAttributes(index) );
    if ( !a.isVisible() ) return;

    // handle decimal digits
    int decimalDigits = a.decimalDigits();
    int decimalPos = QString::number(  value ).indexOf( QLatin1Char( '.' ) );
    QString roundedValue;
    if ( a.dataLabel().isNull() ) {
        if ( decimalPos > 0 && value != 0 )
            roundedValue =  roundValues ( value, decimalPos, decimalDigits );
        else
            roundedValue = QString::number(  value );
    } else
        roundedValue = a.dataLabel();
        // handle prefix and suffix
    if ( !a.prefix().isNull() )
        roundedValue.prepend( a.prefix() );

    if ( !a.suffix().isNull() )
        roundedValue.append( a.suffix() );

    const TextAttributes ta( a.textAttributes() );
    // FIXME draw the non-text bits, background, etc
    if ( ta.isVisible() ) {

        QPointF pt( pos );
        /* for debugging:
        PainterSaver painterSaver( painter );
        painter->setPen( Qt::black );
        painter->drawLine( pos - QPointF( 1,1), pos + QPointF( 1,1) );
        painter->drawLine( pos - QPointF(-1,1), pos + QPointF(-1,1) );
        */

        // adjust the text start point position, if alignment is not Bottom/Left
        const RelativePosition relPos( a.position( value >= 0.0 ) );
        const Qt::Alignment alignBottomLeft = Qt::AlignBottom | Qt::AlignLeft;
        const QFont calculatedFont( ta.calculatedFont( d->plane, KDChartEnums::MeasureOrientationMinim
        //qDebug() << "calculatedFont's point size:" << calculatedFont.pointSizeF();
        if( (relPos.alignment() & alignBottomLeft) != alignBottomLeft ){
            const QRectF boundRect(
                    d->cachedFontMetrics( calculatedFont, this )->boundingRect( roundedValue ) );
            if( relPos.alignment() & Qt::AlignRight )
                pt.rx() -= boundRect.width();
            else if( relPos.alignment() & Qt::AlignHCenter )
                pt.rx() -= 0.5 * boundRect.width();

            if( relPos.alignment() & Qt::AlignTop )
                pt.ry() += boundRect.height();
            else if( relPos.alignment() & Qt::AlignVCenter )
                pt.ry() += 0.5 * boundRect.height();
        }

        // FIXME draw the non-text bits, background, etc

        if ( a.showRepetitiveDataLabels() ||
             pos.x() <= d->lastX ||
             d->lastRoundedValue != roundedValue ) {
            d->lastRoundedValue = roundedValue;
            d->lastX = pos.x();

            PainterSaver painterSaver( painter );
            painter->setPen( ta.pen() );
            painter->setFont( calculatedFont );
            painter->translate( pt );
            painter->rotate( ta.rotation() );
            painter->drawText( QPointF(0, 0), roundedValue );
        }
    }
}


```

7.46.2.41 void AbstractDiagram::paintDataValueTexts (QPainter ∗ *painter*) `[protected, virtual, inherited]`

Definition at line 576 of file KDChartAbstractDiagram.cpp.

```
                                                                               {
       for ( int j=0; j< rowCount; ++j ) {
           const QModelIndex index = model()->index( j, i, rootIndex() );
           double value = model()->data( index ).toDouble();
           const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
           paintDataValueText( painter, index, pos, value );
       }
    }
}


```

7.46.2.42 void PolarDiagram::paintEvent (QPaintEvent ∗) `[protected]`

Definition at line 121 of file KDChartPolarDiagram.cpp.

References paint(), KDChart::PaintContext::setPainter(), and KDChart::PaintContext::setRectangle().

```
{
    QPainter painter ( viewport() );
    PaintContext ctx;
    ctx.setPainter ( &painter );
    ctx.setRectangle( QRectF ( 0, 0, width(), height() ) );
    paint ( &ctx );
}
```

7.46.2.43 void AbstractDiagram::paintMarker (QPainter ∗ *painter*, const QModelIndex & *index*, const QPointF & *pos*) `[inherited]`

Definition at line 592 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::brush(), KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractDiagram::dataValueAttributes(), KDChart::MarkerAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::DataValueAttributes::markerAttributes(), KDChart::MarkerAttributes::markerColor(), KDChart::MarkerAttributes::markerSize(), KDChart::AbstractDiagram::paintMarker(), and KDChart::MarkerAttributes::pen().

```
{

    if ( !checkInvariants() ) return;
    DataValueAttributes a = dataValueAttributes(index);
    if ( !a.isVisible() ) return;
    const MarkerAttributes &ma = a.markerAttributes();
    if ( !ma.isVisible() ) return;

    PainterSaver painterSaver( painter );
    QSizeF maSize( ma.markerSize() );
    QBrush indexBrush( brush( index ) );
    QPen indexPen( ma.pen() );
    if ( ma.markerColor().isValid() )
        indexBrush.setColor( ma.markerColor() );

    paintMarker( painter, ma, indexBrush, indexPen, pos, maSize );
}
```

7.46.2.44 void AbstractDiagram::paintMarker (QPainter ∗ *painter*, const MarkerAttributes & *markerAttributes*, const QBrush & *brush*, const QPen &, const QPointF & *point*, const QSizeF & *size*) `[virtual, inherited]`

Definition at line 614 of file KDChartAbstractDiagram.cpp.

References KDChart::MarkerAttributes::markerStyle().

Referenced by KDChart::MarkerLayoutItem::paintIntoRect(), and KDChart::AbstractDiagram::paintMarker().

```
{

    const QPen oldPen( painter->pen() );
    // Pen is used to paint 4Pixels - 1 Pixel - Ring and FastCross types.
    // make sure to use the brush color - see above in those cases.
    const bool isFourPixels = (markerAttributes.markerStyle() == MarkerAttributes::Marker4Pixels);
    if( isFourPixels || (markerAttributes.markerStyle() == MarkerAttributes::Marker1Pixel) ){
        // for high-performance point charts with tiny point markers:
        painter->setPen( QPen( brush.color().light() ) );
        if( isFourPixels ){
            const qreal x = pos.x();
            const qreal y = pos.y();
            painter->drawLine( QPointF(x-1.0,y-1.0),
                               QPointF(x+1.0,y-1.0) );
            painter->drawLine( QPointF(x-1.0,y),
                               QPointF(x+1.0,y) );
            painter->drawLine( QPointF(x-1.0,y+1.0),
                               QPointF(x+1.0,y+1.0) );
        }
        painter->drawPoint( pos );
    }else{
        PainterSaver painterSaver( painter );
        // we only a solid line surrounding the markers
        QPen painterPen( pen );
        painterPen.setStyle( Qt::SolidLine );
        painter->setPen( painterPen );
        painter->setBrush( brush );
        painter->setRenderHint ( QPainter::Antialiasing );
        painter->translate( pos );
        switch ( markerAttributes.markerStyle() ) {
            case MarkerAttributes::MarkerCircle:
                painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                            maSize.height(), maSize.width()) );
                break;
            case MarkerAttributes::MarkerSquare:
                {
                    QRectF rect( 0 - maSize.width()/2, 0 - maSize.height()/2,
                                maSize.width(), maSize.height() );
                    painter->drawRect( rect );
                    painter->fillRect( rect, brush.color() );
                    break;
                }
            case MarkerAttributes::MarkerDiamond:
                {
                    QVector <QPointF > diamondPoints;
                    QPointF top, left, bottom, right;
                    top    = QPointF( 0, 0 - maSize.height()/2 );
                    left   = QPointF( 0 - maSize.width()/2, 0 );
                    bottom = QPointF( 0, maSize.height()/2 );
                    right  = QPointF( maSize.width()/2, 0 );
                    diamondPoints << top << left << bottom << right;
                    painter->drawPolygon( diamondPoints );
                    break;
                }
            // both handled on top of the method:
            case MarkerAttributes::Marker1Pixel:
            case MarkerAttributes::Marker4Pixels:
                    break;
            case MarkerAttributes::MarkerRing:
                {
                    painter->setPen( QPen( brush.color() ) );
                    painter->setBrush( Qt::NoBrush );
                    painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                                        maSize.height(), maSize.width()) );
                    break;
                }
            case MarkerAttributes::MarkerCross:
                {
                    QRectF rect( maSize.width()*-0.5, maSize.height()*-0.2,
                                 maSize.width(), maSize.height()*0.4 );
                    painter->drawRect( rect );
                    rect.setTopLeft(QPointF( maSize.width()*-0.2, maSize.height()*-0.5 ));
                    rect.setSize(QSizeF( maSize.width()*0.4, maSize.height() ));
                    painter->drawRect( rect );
                    break;
                }
            case MarkerAttributes::MarkerFastCross:
                {
                    QPointF left, right, top, bottom;
                    left  = QPointF( -maSize.width()/2, 0 );
                    right = QPointF( maSize.width()/2, 0 );
                    top   = QPointF( 0, -maSize.height()/2 );
                    bottom= QPointF( 0, maSize.height()/2 );
                    painter->setPen( QPen( brush.color() ) );
                    painter->drawLine( left, right );
                    painter->drawLine(  top, bottom );
                    break;
                }
            default:
                Q_ASSERT_X ( false, "paintMarkers()",
                            "Type item does not match a defined Marker Type." );
        }
```

```
    }
    painter->setPen( oldPen );
}

void AbstractDiagram::paintMarkers( QPainter* painter )
```

7.46.2.45 void AbstractDiagram::paintMarkers (QPainter ∗ *painter*) `[protected, virtual, inherited]`

Definition at line 716 of file KDChartAbstractDiagram.cpp.

```
                                                                                {
       for ( int j=0; j< rowCount; ++j ) {
           const QModelIndex index = model()->index( j, i, rootIndex() );
           double value = model()->data( index ).toDouble();
           const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
           paintMarker( painter, index, pos );
       }
    }
}


```

7.46.2.46 void PolarDiagram::paintPolarMarkers (PaintContext ∗ *ctx*, const QPolygonF & *polygon*) `[protected, virtual]`

Definition at line 134 of file KDChartPolarDiagram.cpp.

References KDChart::PaintContext::painter().

Referenced by paint().

```
{
    const double markerSize = 4; // FIXME use real markers
    for ( int i=0; i<polygon.size(); ++i ) {
        QPointF p = polygon.at( i );
        p.setX( p.x() - markerSize/2 );
        p.setY( p.y() - markerSize/2 );
        ctx->painter()->drawRect( QRectF( p, QSizeF( markerSize, markerSize ) ) );
    }
}
```

7.46.2.47 QPen AbstractDiagram::pen (const QModelIndex & *index*) const `[inherited]`

Retrieve the pen to be used, for painting the datapoint at the given index in the model.

Parameters:

index The index of the datapoint in the model.

Returns:

The pen to use for painting.

Definition at line 770 of file KDChartAbstractDiagram.cpp.

```
{
```

7.46.2.48 QPen AbstractDiagram::pen (int *dataset*) const `[inherited]`

Retrieve the pen to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the pen for.

Returns:

The pen to use for painting.

Definition at line 762 of file KDChartAbstractDiagram.cpp.

```
769 {
```

7.46.2.49 QPen AbstractDiagram::pen () const `[inherited]`

Retrieve the pen to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The pen to use for painting.

Definition at line 756 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::PieDiagram::paint(), and KDChart::LineDiagram::paint().

```
761 {
```

7.46.2.50 bool AbstractDiagram::percentMode () const `[inherited]`

Definition at line 468 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::CartesianCoordinatePlane::getDataDimensionsList().

7.46.2.51 const PolarCoordinatePlane $*$ AbstractPolarDiagram::polarCoordinatePlane () const `[inherited]`

Definition at line 55 of file KDChartAbstractPolarDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane().

Referenced by KDChart::PieDiagram::paint().

```
56 {
57     return dynamic_cast<const PolarCoordinatePlane*>( coordinatePlane() );
58 }
```

7.46.2.52 void KDChart::AbstractDiagram::propertiesChanged () `[protected, inherited]`

Emitted upon change of a property of the Diagram.

Referenced by KDChart::LineDiagram::resetLineAttributes(), KDChart::AbstractDiagram::setDataValueAttributes(), KDChart::LineDiagram::setLineAttributes(), KDChart::LineDiagram::setThreeDLineAttributes(), and KDChart::LineDiagram::setType().

7.46.2.53 void PolarDiagram::resize (const QSizeF & *area*) `[virtual]`

[reimplemented]

Implements KDChart::AbstractDiagram.

Definition at line 191 of file KDChartPolarDiagram.cpp.

```
192 {
193 }
```

7.46.2.54 void PolarDiagram::resizeEvent (QResizeEvent ∗) `[protected]`

Definition at line 130 of file KDChartPolarDiagram.cpp.

```
131 {
132 }
```

7.46.2.55 bool PolarDiagram::rotateCircularLabels () const

Definition at line 229 of file KDChartPolarDiagram.cpp.

References d.

Referenced by clone().

```
230 {
231     return d->rotateCircularLabels;
232 }
```

7.46.2.56 void AbstractDiagram::scrollTo (const QModelIndex & *index*, ScrollHint *hint* = EnsureVisible) `[virtual, inherited]`

[reimplemented]

Definition at line 830 of file KDChartAbstractDiagram.cpp.

```
832 { return QModelIndex(); }
```

7.46.2.57 void AbstractDiagram::setAllowOverlappingDataValueTexts (bool *allow*) `[inherited]`

Set whether data value labels are allowed to overlap.

Parameters:

allow True means that overlapping labels are allowed.

Definition at line 440 of file KDChartAbstractDiagram.cpp.

References d.

```
445 {
```

7.46.2.58 void AbstractDiagram::setAntiAliasing (bool *enabled*) `[inherited]`

Set whether anti-aliasing is to be used while rendering this diagram.

Parameters:

enabled True means that AA is enabled.

Definition at line 451 of file KDChartAbstractDiagram.cpp.

References d.

```
456 {
```

7.46.2.59 void AbstractDiagram::setAttributesModel (AttributesModel ∗ *model*) `[virtual, inherited]`

Associate an AttributesModel with this diagram.

Note that the diagram does _not_ take ownership of the AttributesModel. This should thus only be used with AttributesModels that have been explicitely created by the user, and are owned by her. Setting an AttributesModel that is internal to another diagram is an error.

Correct:

```
AttributesModel *am = new AttributesModel( model, 0 );
diagram1->setAttributesModel( am );
diagram2->setAttributesModel( am );
```

Wrong:

```
diagram1->setAttributesModel( diagram2->attributesModel() );
```

Parameters:

model The AttributesModel to use for this diagram.

See also:

AttributesModel, usesExternalAttributesModel

Definition at line 261 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::AbstractDiagram::modelsChanged().

```
262 {
263     if( amodel->sourceModel() != model() ) {
264         qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
265                  "Trying to set an attributesmodel which works on a different "
266                  "model than the diagram.");
267         return;
268     }
269     if( qobject_cast<PrivateAttributesModel*>(amodel) ) {
270         qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
271                  "Trying to set an attributesmodel that is private to another diagram.");
272         return;
273     }
274     d->setAttributesModel(amodel);
275     scheduleDelayedItemsLayout();
276     d->databoundariesDirty = true;
277     emit modelsChanged();
278 }
```

7.46.2.60 void AbstractDiagram::setAttributesModelRootIndex (const QModelIndex & *idx*) `[protected, inherited]`

Definition at line 301 of file KDChartAbstractDiagram.cpp.

References d.

7.46.2.61 void AbstractDiagram::setBrush (const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting all datasets in the model.

Parameters:

brush The brush to use.

Definition at line 786 of file KDChartAbstractDiagram.cpp.

```
792 {
```

7.46.2.62 void AbstractDiagram::setBrush (int *dataset*, const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting the given dataset.

Parameters:

dataset The dataset's column in the model.

pen The brush to use.

Definition at line 793 of file KDChartAbstractDiagram.cpp.

```
801 {
```

7.46.2.63 void AbstractDiagram::setBrush (const QModelIndex & *index*, const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting the datapoint at the given index.

Parameters:

index The datapoint's index in the model.

brush The brush to use.

Definition at line 778 of file KDChartAbstractDiagram.cpp.

```
785 {
```

7.46.2.64 void PolarDiagram::setCloseDatasets (bool *closeDatasets*)

Close each of the data series by connecting the last point to its respective start point.

Definition at line 234 of file KDChartPolarDiagram.cpp.

References d.

```
235 {
236     d->closeDatasets = closeDatasets;
237 }
```

7.46.2.65 void AbstractDiagram::setCoordinatePlane (AbstractCoordinatePlane ∗ *plane*) `[virtual, inherited]`

Set the coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. The chart takes ownership.

Returns:

The coordinate plane associated with the diagram.

Reimplemented in KDChart::AbstractCartesianDiagram.

Definition at line 324 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractCoordinatePlane::addDiagram(), KDChart::AbstractCartesianDiagram::setCoordinatePlane(), and KDChart::AbstractCoordinatePlane::takeDiagram().

```
328 {
```

7.46.2.66 void AbstractDiagram::setDataBoundariesDirty () const `[protected, inherited]`

Definition at line 240 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::BarDiagram::setThreeDBarAttributes(), KDChart::LineDiagram::setThreeDLineAttributes(), KDChart::LineDiagram::setType(), and KDChart::BarDiagram::setType().

```
241 {
242     d->databoundariesDirty = true;
243 }
```

7.46.2.67 void AbstractDiagram::setDatasetDimension (int *dimension*) `[inherited]`

Sets the dataset dimension of the diagram.

See also:
datasetDimension.

Parameters:
dimension

Definition at line 947 of file KDChartAbstractDiagram.cpp.

References d.

```
954 {
```

7.46.2.68 void AbstractDiagram::setDataValueAttributes (const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for all datapoints in the model.

Parameters:
a The attributes to set.

Definition at line 434 of file KDChartAbstractDiagram.cpp.

References d.

```
439 {
```

7.46.2.69 void AbstractDiagram::setDataValueAttributes (int *dataset*, const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for the given dataset.

Parameters:
dataset The dataset to set the attributes for.
a The attributes to set.

Definition at line 406 of file KDChartAbstractDiagram.cpp.

References d.

```
413 {
```

7.46.2.70 void AbstractDiagram::setDataValueAttributes (const QModelIndex & *index*, const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for the given index.

Parameters:

index The datapoint to set the attributes for.

a The attributes to set.

Definition at line 395 of file KDChartAbstractDiagram.cpp.

References d, KDChart::DataValueLabelAttributesRole, and KDChart::AbstractDiagram::propertiesChanged().

```
395 {
396     d->attributesModel->setData(
397         d->attributesModel->mapFromSource( index ),
398         qVariantFromValue( a ),
399         DataValueLabelAttributesRole );
400     emit propertiesChanged();
401 }
402
403
```

7.46.2.71 void AbstractDiagram::setHidden (bool *hidden*) `[inherited]`

Hide (or unhide, resp.) all datapoints in the model.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

hidden The hidden status to set.

Definition at line 365 of file KDChartAbstractDiagram.cpp.

References d.

```
372 {
```

7.46.2.72 void AbstractDiagram::setHidden (int *column*, bool *hidden*) `[inherited]`

Hide (or unhide, resp.) a dataset.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

dataset The dataset to set the hidden status for.

hidden The hidden status to set.

Definition at line 356 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.46.2.73 void AbstractDiagram::setHidden (const QModelIndex & *index*, bool *hidden*) `[inherited]`

Hide (or unhide, resp.) a data cell.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

index The datapoint to set the hidden status for.

hidden The hidden status to set.

Definition at line 347 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::DataHiddenRole.

```
{
```

7.46.2.74 void AbstractDiagram::setModel (QAbstractItemModel ∗ *model*) `[virtual, inherited]`

Associate a model with the diagram.

Definition at line 245 of file KDChartAbstractDiagram.cpp.

References d, KDChart::AttributesModel::initFrom(), and KDChart::AbstractDiagram::modelsChanged().

```
{
  QAbstractItemView::setModel( newModel );
  AttributesModel* amodel = new PrivateAttributesModel( newModel, this );
  amodel->initFrom( d->attributesModel );
  d->setAttributesModel(amodel);
  scheduleDelayedItemsLayout();
  d->databoundariesDirty = true;
  emit modelsChanged();
}
```

7.46.2.75 void AbstractDiagram::setPen (const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting all datasets in the model.

Parameters:

pen The pen to use.

Definition at line 740 of file KDChartAbstractDiagram.cpp.

```
746 {
```

7.46.2.76 void AbstractDiagram::setPen (int *dataset*, const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting the given dataset.

Parameters:

dataset The dataset's row in the model.

pen The pen to use.

Definition at line 747 of file KDChartAbstractDiagram.cpp.

```
755 {
```

7.46.2.77 void AbstractDiagram::setPen (const QModelIndex & *index*, const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting the datapoint at the given index.

Parameters:

index The datapoint's index in the model.

pen The pen to use.

Definition at line 732 of file KDChartAbstractDiagram.cpp.

```
739 {
```

7.46.2.78 void AbstractDiagram::setPercentMode (bool *percent*) `[inherited]`

Definition at line 462 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::setType(), and KDChart::BarDiagram::setType().

```
467 {
```

7.46.2.79 void AbstractDiagram::setRootIndex (const QModelIndex & *idx*) `[virtual, inherited]`

Set the root index in the model, where the diagram starts referencing data for display.

[reimplemented]

Definition at line 294 of file KDChartAbstractDiagram.cpp.

References d.

7.46.2.80 void PolarDiagram::setRotateCircularLabels (bool *rotateCircularLabels*)

Definition at line 224 of file KDChartPolarDiagram.cpp.

References d.

```
{
    d->rotateCircularLabels = rotateCircularLabels;
}
```

7.46.2.81 void AbstractDiagram::setSelection (const QRect & *rect*, QItemSelectionModel::SelectionFlags *command*) `[virtual, inherited]`

[reimplemented]

Definition at line 848 of file KDChartAbstractDiagram.cpp.

```
{ return QRegion(); }
```

7.46.2.82 void PolarDiagram::setShowDelimitersAtPosition (Position *position*, bool *showDelimiters*)

Definition at line 244 of file KDChartPolarDiagram.cpp.

References d, and KDChart::Position::value().

```
{
    d->showDelimitersAtPosition[position.value()] = showDelimiters;
}
```

7.46.2.83 void PolarDiagram::setShowLabelsAtPosition (Position *position*, bool *showLabels*)

Definition at line 250 of file KDChartPolarDiagram.cpp.

References d, and KDChart::Position::value().

```
{
    d->showLabelsAtPosition[position.value()] = showLabels;
}
```

7.46.2.84 void PolarDiagram::setZeroDegreePosition (int *degrees*)

Deprecated

Use PolarCoordinatePlane::setStartPosition(qreal degrees) instead.

Definition at line 213 of file KDChartPolarDiagram.cpp.

```
{
    qWarning() << "Deprecated PolarDiagram::setZeroDegreePosition() called, setting ignored.";
}
```

7.46.2.85 bool PolarDiagram::showDelimitersAtPosition (Position *position*) const

Definition at line 256 of file KDChartPolarDiagram.cpp.

References d, and KDChart::Position::value().

Referenced by clone().

```
{
    return d->showDelimitersAtPosition[position.value()];
}
```

7.46.2.86 bool PolarDiagram::showLabelsAtPosition (Position *position*) const

Definition at line 261 of file KDChartPolarDiagram.cpp.

References d, and KDChart::Position::value().

Referenced by clone().

```
{
    return d->showLabelsAtPosition[position.value()];
}
```

7.46.2.87 void AbstractDiagram::update () const `[inherited]`

Definition at line 961 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractDiagram::doItemsLayout().

7.46.2.88 void KDChart::AbstractDiagram::useDefaultColors () `[inherited]`

Set the palette to be used, for painting datasets to the default palette.

See also:

KDChart::Palette. FIXME: fold into one usePalette (KDChart::Palette&) method

Definition at line 855 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.46.2.89 void KDChart::AbstractDiagram::useRainbowColors () `[inherited]`

Set the palette to be used, for painting datasets to the rainbow palette.

See also:
KDChart::Palette.

Definition at line 865 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.46.2.90 bool AbstractDiagram::usesExternalAttributesModel () const `[virtual, inherited]`

Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

See also:
setAttributesModel

Definition at line 280 of file KDChartAbstractDiagram.cpp.

References d.

```
{
    return d->usesExternalAttributesModel();
}
```

7.46.2.91 void KDChart::AbstractDiagram::useSubduedColors () `[inherited]`

Set the palette to be used, for painting datasets to the subdued palette.

See also:
KDChart::Palette.

Definition at line 860 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.46.2.92 double AbstractDiagram::valueForCell (int *row*, int *column*) const `[protected, inherited]`

Helper method, retrieving the data value (DisplayRole) for a given row and column.

Parameters:
row The row to query.

column The column to query.

Returns:

The value of the display role at the given row and column as a double.

Definition at line 955 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::attributesModelRootIndex(), and d.

Referenced by KDChart::LineDiagram::paint().

```
{
```

7.46.2.93 double PolarDiagram::valueTotals () const `[virtual]`

[reimplemented]

Implements KDChart::AbstractPolarDiagram.

Definition at line 196 of file KDChartPolarDiagram.cpp.

```
{
    return model()->rowCount(rootIndex());
}
```

7.46.2.94 int AbstractDiagram::verticalOffset () const `[virtual, inherited]`

[reimplemented]

Definition at line 842 of file KDChartAbstractDiagram.cpp.

```
{ return true; }
```

7.46.2.95 QRect AbstractDiagram::visualRect (const QModelIndex & *index*) const `[virtual, inherited]`

[reimplemented]

Definition at line 825 of file KDChartAbstractDiagram.cpp.

```
{}
```

7.46.2.96 QRegion AbstractDiagram::visualRegionForSelection (const QItemSelection & *selection*) const `[virtual, inherited]`

[reimplemented]

Definition at line 851 of file KDChartAbstractDiagram.cpp.

7.46.2.97 int PolarDiagram::zeroDegreePosition () const

Deprecated

Use qreal PolarCoordinatePlane::startPosition instead.

Definition at line 218 of file KDChartPolarDiagram.cpp.

```
{
    qWarning() << "Deprecated PolarDiagram::zeroDegreePosition() called.";
    return 0;
}
```

7.46.3 Member Data Documentation

7.46.3.1 Q_SIGNALS KDChart::AbstractDiagram::__pad0__ `[protected, inherited]`

Definition at line 589 of file KDChartAbstractDiagram.h.

The documentation for this class was generated from the following files:

- KDChartPolarDiagram.h
- KDChartPolarDiagram.cpp

7.47 KDChart::Position Class Reference

`#include <KDChartPosition.h>`

Collaboration diagram for KDChart::Position:

7.47.1 Detailed Description

Defines a position, using compass terminology.

Using KDChartPosition you can specify one of nine pre-defined, logical points (see the `static const` getter methods below), in a similar way, as you would use a compass to navigate on a map.

Often you will declare a `Position` together with the RelativePosition class, to specify a logical point, which then will be used to layout your chart at runtime, e.g. for specifying the location of a floating Legend box.

For comparing a Position's value with a switch() statement, you can use numeric values defined in KDChartEnums, like this:

```
switch( yourPosition().value() ) {
    case KDChartEnums::PositionNorthWest:
        // your code ...
        break;
    case KDChartEnums::PositionNorth:
        // your code ...
        break;
}
```

See also:
RelativePosition, KDChartEnums::PositionValue

Definition at line 75 of file KDChartPosition.h.

Public Types

- enum Option {
  IncludeCenter = 0,
  ExcludeCenter = 1 }

Public Member Functions

- bool isCorner () const
- bool isEastSide () const
- bool isFloating () const
- bool isNorthSide () const
- bool isPole () const
- bool isSouthSide () const
- bool isUnknown () const
- bool isWestSide () const
- const char ∗ name () const

  Returns a non-translated string in English language, corresponding to this Position.

- bool operator!= (int) const
- bool operator!= (const Position &) const
- bool operator== (int) const
- bool operator== (const Position &) const
- Position (KDChartEnums::PositionValue value)

  Constructor.

- Position ()

  Default constructor.

- QString printableName () const

  Returns a translated string, corresponding to this Position.

- KDChartEnums::PositionValue value () const

  Returns an integer value corresponding to this Position.

Static Public Member Functions

- Position fromName (const QByteArray &name)
- Position fromName (const char ∗name)
- QList< QByteArray > names (Options options=IncludeCenter)

  Returns a list of all string, corresponding to the pre-defined positions.

- QStringList printableNames (Options options=IncludeCenter)

  Returns a list of all translated string, corresponding to the pre-defined positions.

Static Public Attributes

- const Position & Center = staticPositionCenter
- const Position & East = staticPositionEast
- const Position & Floating = staticPositionFloating
- const Position & North = staticPositionNorth
- const Position & NorthEast = staticPositionNorthEast
- const Position & NorthWest = staticPositionNorthWest
- const Position & South = staticPositionSouth
- const Position & SouthEast = staticPositionSouthEast
- const Position & SouthWest = staticPositionSouthWest
- const Position & Unknown = staticPositionUnknown
- const Position & West = staticPositionWest

7.47.2 Member Enumeration Documentation

7.47.2.1 enum KDChart::Position::Option

Enumeration values:

IncludeCenter

ExcludeCenter

Definition at line 113 of file KDChartPosition.h.

```
113 { IncludeCenter=0, ExcludeCenter=1 };
```

7.47.3 Constructor & Destructor Documentation

7.47.3.1 Position::Position ()

Default constructor.

Creates a new Position, defaulting it to Position::Unknown.

Definition at line 100 of file KDChartPosition.cpp.

Referenced by fromName(), and printableNames().

```
101     : m_value( KDChartEnums::PositionUnknown )
102 {
103
104 }
```

7.47.3.2 Position::Position (KDChartEnums::PositionValue *value*)

Constructor.

Creates a new Position, defaulting it to the respective value.

Valid values ranging from zero (unknown value) to 10. If invalid value is passed, a Position::Unknown is created.

Note:

Normally there is no need to call this constructor, but you would rather use one of the nine pre-defined, static values, e.g. like this:

```
 * const KDChart::Position myPosition = KDChart::Position::NorthEast;
 *
```

Definition at line 124 of file KDChartPosition.cpp.

```
125     : m_value( value )
126 {
127
128 }
```

7.47.4 Member Function Documentation

7.47.4.1 Position Position::fromName (const QByteArray & *name*) `[static]`

Definition at line 243 of file KDChartPosition.cpp.

References fromName().

```
243                                                      {
244     return fromName( name.data() );
245 }
```

7.47.4.2 Position Position::fromName (const char ∗ *name*) `[static]`

Definition at line 235 of file KDChartPosition.cpp.

References maxPositionValue, Position(), and staticPositionNames.

Referenced by fromName().

```
{
    for( int i=1; i<=maxPositionValue; ++i)
        if ( !qstricmp( name, staticPositionNames[i] ) )
            return Position(i);
    return Position(0);
}
```

7.47.4.3 bool Position::isCorner () const

Definition at line 168 of file KDChartPosition.cpp.

References value().

```
{
    return  m_value == Position::NorthWest.value() ||
            m_value == Position::NorthEast.value() ||
            m_value == Position::SouthEast.value() ||
            m_value == Position::SouthWest.value();
}
```

7.47.4.4 bool Position::isEastSide () const

Definition at line 155 of file KDChartPosition.cpp.

References value().

```
{
    return  m_value == Position::NorthEast.value() ||
            m_value == Position::East.value() ||
            m_value == Position::SouthEast.value();
}
```

7.47.4.5 bool Position::isFloating () const

Definition at line 181 of file KDChartPosition.cpp.

References value().

Referenced by KDChart::Chart::reLayoutFloatingLegends().

```
{
    return  m_value == Position::Floating.value();
}
```

7.47.4.6 bool Position::isNorthSide () const

Definition at line 149 of file KDChartPosition.cpp.

References value().

```
150 {
151     return  m_value == Position::NorthWest.value() ||
152             m_value == Position::North.value() ||
153             m_value == Position::NorthEast.value();
154 }
```

7.47.4.7 bool Position::isPole () const

Definition at line 175 of file KDChartPosition.cpp.

References value().

```
176 {
177     return  m_value == Position::North.value() ||
178         m_value == Position::South.value();
179 }
```

7.47.4.8 bool Position::isSouthSide () const

Definition at line 161 of file KDChartPosition.cpp.

References value().

```
162 {
163     return  m_value == Position::SouthWest.value() ||
164             m_value == Position::South.value() ||
165             m_value == Position::SouthEast.value();
166 }
```

7.47.4.9 bool Position::isUnknown () const

Definition at line 138 of file KDChartPosition.cpp.

References value().

```
139 {
140     return  m_value == Position::Unknown.value();
141 }
```

7.47.4.10 bool Position::isWestSide () const

Definition at line 143 of file KDChartPosition.cpp.

References value().

```
144 {
145     return  m_value == Position::SouthWest.value() ||
146             m_value == Position::West.value() ||
147             m_value == Position::NorthWest.value();
148 }
```

7.47.4.11 const char ∗ Position::name () const

Returns a non-translated string in English language, corresponding to this Position.

Definition at line 189 of file KDChartPosition.cpp.

References staticPositionNames.

Referenced by operator<<().

```
190 {
191     return staticPositionNames[m_value];
192 }
```

7.47.4.12 QList< QByteArray > Position::names (Options *options* = IncludeCenter) `[static]`

Returns a list of all string, corresponding to the pre-defined positions.

Parameters:

options if set to `ExcludeCenter`, the returned list does not contain the Center position.

Definition at line 210 of file KDChartPosition.cpp.

References IncludeCenter, maxPositionValue, and staticPositionNames.

```
211 {
212     QList<QByteArray> list;
213     const int start = ( options & IncludeCenter ) ? 1 : 2;
214     for( int i=start; i<=maxPositionValue; ++i)
215         list.append( staticPositionNames[i] );
216     return list;
217 }
```

7.47.4.13 bool KDChart::Position::operator!= (int) const

Definition at line 132 of file KDChartPosition.h.

References operator==().

```
132 { return !operator==( other ); }
```

7.47.4.14 bool KDChart::Position::operator!= (const Position &) const

Definition at line 131 of file KDChartPosition.h.

References operator==().

```
131 { return !operator==( other ); }
```

7.47.4.15 bool Position::operator== (int) const

Definition at line 253 of file KDChartPosition.cpp.

References value().

```
{
    return ( value() == value_ );
}
```

7.47.4.16 bool Position::operator== (const Position &) const

Definition at line 247 of file KDChartPosition.cpp.

References value().

Referenced by operator!=().

```
{
    return ( value() == r.value() );
}
```

7.47.4.17 QString Position::printableName () const

Returns a translated string, corresponding to this Position.

Definition at line 197 of file KDChartPosition.cpp.

References staticPositionNames.

Referenced by printableNames().

```
{
    return tr(staticPositionNames[m_value]);
}
```

7.47.4.18 QStringList Position::printableNames (Options *options* = IncludeCenter) `[static]`

Returns a list of all translated string, corresponding to the pre-defined positions.

Parameters:

options if set to `ExcludeCenter`, the returned list does not contain the Center position.

Definition at line 226 of file KDChartPosition.cpp.

References IncludeCenter, maxPositionValue, Position(), and printableName().

```
{
    QStringList list;
    const int start = ( options & IncludeCenter ) ? 1 : 2;
    for( int i=start; i<=maxPositionValue; ++i)
        list.append( Position(i).printableName() );
    return list;
}
```

7.47.4.19 KDChartEnums::PositionValue Position::value () const

Returns an integer value corresponding to this Position.

Definition at line 133 of file KDChartPosition.cpp.

Referenced by isCorner(), isEastSide(), isFloating(), isNorthSide(), isPole(), isSouthSide(), isUnknown(), isWestSide(), operator==(), KDChart::PolarDiagram::setShowDelimitersAtPosition(), KDChart::PolarDiagram::setShowLabelsAtPosition(), KDChart::PolarDiagram::showDelimitersAtPosition(), and KDChart::PolarDiagram::showLabelsAtPosition().

```
{
    return static_cast<KDChartEnums::PositionValue>( m_value );
}
```

7.47.5 Member Data Documentation

7.47.5.1 const Position & Position::Center = staticPositionCenter `[static]`

Definition at line 85 of file KDChartPosition.cpp.

7.47.5.2 const Position & Position::East = staticPositionEast `[static]`

Definition at line 89 of file KDChartPosition.cpp.

7.47.5.3 const Position & Position::Floating = staticPositionFloating `[static]`

Definition at line 94 of file KDChartPosition.cpp.

7.47.5.4 const Position & Position::North = staticPositionNorth `[static]`

Definition at line 87 of file KDChartPosition.cpp.

7.47.5.5 const Position & Position::NorthEast = staticPositionNorthEast `[static]`

Definition at line 88 of file KDChartPosition.cpp.

7.47.5.6 const Position & Position::NorthWest = staticPositionNorthWest `[static]`

Definition at line 86 of file KDChartPosition.cpp.

7.47.5.7 const Position & Position::South = staticPositionSouth `[static]`

Definition at line 91 of file KDChartPosition.cpp.

7.47.5.8 const Position & Position::SouthEast = staticPositionSouthEast `[static]`

Definition at line 90 of file KDChartPosition.cpp.

7.47.5.9 const Position & Position::SouthWest = staticPositionSouthWest `[static]`

Definition at line 92 of file KDChartPosition.cpp.

7.47.5.10 const Position & Position::Unknown = staticPositionUnknown `[static]`

Definition at line 84 of file KDChartPosition.cpp.

7.47.5.11 const Position & Position::West = staticPositionWest `[static]`

Definition at line 93 of file KDChartPosition.cpp.

The documentation for this class was generated from the following files:

- KDChartPosition.h
- KDChartPosition.cpp

7.48 KDChart::PositionPoints Class Reference

`#include <KDChartPosition.h>`

Collaboration diagram for KDChart::PositionPoints:

Public Member Functions

- bool isNull () const
- const QPointF point (Position position) const
- PositionPoints (QPointF northWest, QPointF northEast, QPointF southEast, QPointF southWest)
- PositionPoints (const QRectF &rect)
- PositionPoints (const QPointF &onePointForAllPositions)
- PositionPoints (QPointF center, QPointF northWest, QPointF north, QPointF northEast, QPointF east, QPointF southEast, QPointF south, QPointF southWest, QPointF west)
- PositionPoints ()

Public Attributes

- QPointF mPositionCenter
- QPointF mPositionEast
- QPointF mPositionNorth
- QPointF mPositionNorthEast
- QPointF mPositionNorthWest
- QPointF mPositionSouth
- QPointF mPositionSouthEast
- QPointF mPositionSouthWest
- QPointF mPositionUnknown
- QPointF mPositionWest

7.48.1 Constructor & Destructor Documentation

7.48.1.1 KDChart::PositionPoints::PositionPoints ()

Definition at line 138 of file KDChartPosition.h.

```
138 {} // all points get initialized with the default automatically
```

7.48.1.2 KDChart::PositionPoints::PositionPoints (QPointF *center*, QPointF *northWest*, QPointF *north*, QPointF *northEast*, QPointF *east*, QPointF *southEast*, QPointF *south*, QPointF *southWest*, QPointF *west*)

Definition at line 140 of file KDChartPosition.h.

```
150       : mPositionCenter(    center )
151       , mPositionNorthWest( northWest )
152       , mPositionNorth(     north )
153       , mPositionNorthEast( northEast )
154       , mPositionEast(      east )
155       , mPositionSouthEast( southEast )
```

```
      , mPositionSouth(     south )
      , mPositionSouthWest( southWest )
      , mPositionWest(      west )
        {}
```

7.48.1.3 KDChart::PositionPoints::PositionPoints (const QPointF & *onePointForAllPositions*)

Definition at line 160 of file KDChartPosition.h.

```
      : mPositionCenter(    onePointForAllPositions )
      , mPositionNorthWest( onePointForAllPositions )
      , mPositionNorth(     onePointForAllPositions )
      , mPositionNorthEast( onePointForAllPositions )
      , mPositionEast(      onePointForAllPositions )
      , mPositionSouthEast( onePointForAllPositions )
      , mPositionSouth(     onePointForAllPositions )
      , mPositionSouthWest( onePointForAllPositions )
      , mPositionWest(      onePointForAllPositions )
        {}
```

7.48.1.4 KDChart::PositionPoints::PositionPoints (const QRectF & *rect*)

Definition at line 172 of file KDChartPosition.h.

```
    {
        const QRectF r( rect.normalized() );
        mPositionCenter    = r.center();
        mPositionNorthWest = r.topLeft();
        mPositionNorth     = QPointF(r.center().x(), r.top());
        mPositionNorthEast = r.topRight();
        mPositionEast      = QPointF(r.right(), r.center().y());
        mPositionSouthEast = r.bottomRight();
        mPositionSouth     = QPointF(r.center().x(), r.bottom());
        mPositionSouthWest = r.bottomLeft();
        mPositionWest      = QPointF(r.left(), r.center().y());
    }
```

7.48.1.5 KDChart::PositionPoints::PositionPoints (QPointF *northWest*, QPointF *northEast*, QPointF *southEast*, QPointF *southWest*)

Definition at line 186 of file KDChartPosition.h.

```
      : mPositionCenter(    (northWest + southEast) / 2.0 )
      , mPositionNorthWest( northWest )
      , mPositionNorth(     (northWest + northEast) / 2.0 )
      , mPositionNorthEast( northEast )
      , mPositionEast(      (northEast + southEast) / 2.0 )
      , mPositionSouthEast( southEast )
      , mPositionSouth(     (southWest + southEast) / 2.0 )
      , mPositionSouthWest( southWest )
      , mPositionWest(      (northWest + southWest) / 2.0 )
        {}
```

7.48.2 Member Function Documentation

7.48.2.1 bool KDChart::PositionPoints::isNull () const

Definition at line 226 of file KDChartPosition.h.

```
{
    return
        mPositionUnknown.isNull() &&
        mPositionCenter.isNull() &&
        mPositionNorthWest.isNull() &&
        mPositionNorth.isNull() &&
        mPositionNorthEast.isNull() &&
        mPositionEast.isNull() &&
        mPositionSouthEast.isNull() &&
        mPositionSouth.isNull() &&
        mPositionSouthWest.isNull() &&
        mPositionWest.isNull();
}
```

7.48.2.2 const QPointF KDChart::PositionPoints::point (Position *position*) const

Definition at line 202 of file KDChartPosition.h.

```
{
  //qDebug() << "point( " << position.name() << " )";
  if( position ==  Position::Center)
    return mPositionCenter;
  if( position ==  Position::NorthWest)
    return mPositionNorthWest;
  if( position ==  Position::North)
    return mPositionNorth;
  if( position ==  Position::NorthEast)
    return mPositionNorthEast;
  if( position ==  Position::East)
    return mPositionEast;
  if( position ==  Position::SouthEast)
    return mPositionSouthEast;
  if( position ==  Position::South)
    return mPositionSouth;
  if( position ==  Position::SouthWest)
    return mPositionSouthWest;
  if( position ==  Position::West)
    return mPositionWest;
  return mPositionUnknown;
}
```

7.48.3 Member Data Documentation

7.48.3.1 QPointF KDChart::PositionPoints::mPositionCenter

Definition at line 242 of file KDChartPosition.h.

7.48.3.2 QPointF KDChart::PositionPoints::mPositionEast

Definition at line 246 of file KDChartPosition.h.

7.48.3.3 QPointF KDChart::PositionPoints::mPositionNorth

Definition at line 244 of file KDChartPosition.h.

7.48.3.4 QPointF KDChart::PositionPoints::mPositionNorthEast

Definition at line 245 of file KDChartPosition.h.

7.48.3.5 QPointF KDChart::PositionPoints::mPositionNorthWest

Definition at line 243 of file KDChartPosition.h.

7.48.3.6 QPointF KDChart::PositionPoints::mPositionSouth

Definition at line 248 of file KDChartPosition.h.

7.48.3.7 QPointF KDChart::PositionPoints::mPositionSouthEast

Definition at line 247 of file KDChartPosition.h.

7.48.3.8 QPointF KDChart::PositionPoints::mPositionSouthWest

Definition at line 249 of file KDChartPosition.h.

7.48.3.9 QPointF KDChart::PositionPoints::mPositionUnknown

Definition at line 241 of file KDChartPosition.h.

7.48.3.10 QPointF KDChart::PositionPoints::mPositionWest

Definition at line 250 of file KDChartPosition.h.

The documentation for this class was generated from the following file:

- KDChartPosition.h

7.49 PrerenderedElement Class Reference

`#include <KDChartTextLabelCache.h>`

Inheritance diagram for PrerenderedElement:Collaboration diagram for PrerenderedElement:

Public Member Functions

- virtual const QPixmap & pixmap () const=0

  Returns the rendered element.

- const QPointF & position () const

  Get the position of the element.

- PrerenderedElement ()
- KDChartEnums::PositionValue referencePoint () const

  Get the reference point of the element.

- virtual QPointF referencePointLocation (KDChartEnums::PositionValue) const=0

  Return the location of the reference point relatively to the pixmap's origin.

- void setPosition (const QPointF &position)

  Set the position of the element.

- void setReferencePoint (KDChartEnums::PositionValue)

  Set the reference point of the element.

- virtual ∼PrerenderedElement ()

Protected Member Functions

- virtual void invalidate () const=0

  invalidate() needs to be called if any of the properties that determine the visual appearance of the prerendered element change.

7.49.1 Constructor & Destructor Documentation

7.49.1.1 PrerenderedElement::PrerenderedElement ()

Definition at line 30 of file KDChartTextLabelCache.cpp.

```
31     : m_referencePoint( KDChartEnums::PositionNorthWest )
32 {
33 }
```

7.49.1.2 virtual PrerenderedElement::∼PrerenderedElement () `[virtual]`

Definition at line 13 of file KDChartTextLabelCache.h.

```
13 {}
```

7.49.2 Member Function Documentation

7.49.2.1 virtual void PrerenderedElement::invalidate () const `[protected, pure virtual]`

invalidate() needs to be called if any of the properties that determine the visual appearance of the prerendered element change.

It can be called for a const object, as objects may need to force recalculation of the pixmap.

Implemented in PrerenderedLabel.

7.49.2.2 virtual const QPixmap& PrerenderedElement::pixmap () const `[pure virtual]`

Returns the rendered element.

If any of the properties have change, the element will be regenerated.

Implemented in PrerenderedLabel.

7.49.2.3 const QPointF & PrerenderedElement::position () const

Get the position of the element.

Definition at line 40 of file KDChartTextLabelCache.cpp.

```
{
    return m_position;
}
```

7.49.2.4 KDChartEnums::PositionValue PrerenderedElement::referencePoint () const

Get the reference point of the element.

Definition at line 50 of file KDChartTextLabelCache.cpp.

Referenced by PrerenderedLabel::referencePointLocation().

```
{
    return m_referencePoint;
}
```

7.49.2.5 virtual QPointF PrerenderedElement::referencePointLocation (KDChartEnums::PositionValue) const `[pure virtual]`

Return the location of the reference point relatively to the pixmap's origin.

Implemented in PrerenderedLabel.

7.49.2.6 void PrerenderedElement::setPosition (const QPointF & *position*)

Set the position of the element.

Definition at line 35 of file KDChartTextLabelCache.cpp.

```
{   // this does not invalidate the element
    m_position = position;
}
```

7.49.2.7 void PrerenderedElement::setReferencePoint (KDChartEnums::PositionValue)

Set the reference point of the element.

Every element has nine possible reference points (all compass directions, plus the center.

Definition at line 45 of file KDChartTextLabelCache.cpp.

```
{   // this does not invalidate the element
    m_referencePoint = point;
}
```

The documentation for this class was generated from the following files:

- KDChartTextLabelCache.h
- KDChartTextLabelCache.cpp

7.50 PrerenderedLabel Class Reference

`#include <KDChartTextLabelCache.h>`

Inheritance diagram for PrerenderedLabel:Collaboration diagram for PrerenderedLabel:

7.50.1 Detailed Description

CachedLabel is an internal KDChart class that simplifies creation and caching of cached text labels.

It provides referenze points to anchor the text to other elements. Reference points use the positions defined in KDChartEnums.

Usage:

```
    double angle = 90.0;
    CachedLabel label;
    label.paint( font, tr("Label"), angle );
```

Definition at line 69 of file KDChartTextLabelCache.h.

Public Member Functions

- double angle () const
- const QBrush & brush () const
- const QFont & font () const
- const QPen & pen () const
- const QPixmap & pixmap () const

  Returns the rendered element.

- const QPointF & position () const

  Get the position of the element.

- PrerenderedLabel ()
- KDChartEnums::PositionValue referencePoint () const

  Get the reference point of the element.

- QPointF referencePointLocation () const
- QPointF referencePointLocation (KDChartEnums::PositionValue position) const

  Return the location of the reference point relatively to the pixmap's origin.

- void setAngle (double angle)
- void setBrush (const QBrush &brush)
- void setFont (const QFont &font)
- void setPen (const QPen &)
- void setPosition (const QPointF &position)

  Set the position of the element.

- void setReferencePoint (KDChartEnums::PositionValue)

  Set the reference point of the element.

- void setText (const QString &text)
- const QString & text () const
- ∼PrerenderedLabel ()

Protected Member Functions

- void invalidate () const

  invalidate() needs to be called if any of the properties that determine the visual appearance of the prerendered element change.

7.50.2 Constructor & Destructor Documentation

7.50.2.1 PrerenderedLabel::PrerenderedLabel ()

Definition at line 55 of file KDChartTextLabelCache.cpp.

```
    : PrerenderedElement()
    , m_dirty( true )
    , m_font( qApp->font() )
    , m_brush( Qt::black )
    , m_pen( Qt::black ) // do not use anything invisible
    , m_angle( 0.0 )
{
}
```

7.50.2.2 PrerenderedLabel::∼PrerenderedLabel ()

Definition at line 65 of file KDChartTextLabelCache.cpp.

References DUMP_CACHE_STATS.

```
{
    DUMP_CACHE_STATS;
}
```

7.50.3 Member Function Documentation

7.50.3.1 double PrerenderedLabel::angle () const

Definition at line 114 of file KDChartTextLabelCache.cpp.

```
{
    return m_angle;
}
```

7.50.3.2 const QBrush & PrerenderedLabel::brush () const

Definition at line 103 of file KDChartTextLabelCache.cpp.

```
104 {
105     return m_brush;
106 }
```

7.50.3.3 const QFont & PrerenderedLabel::font () const

Definition at line 81 of file KDChartTextLabelCache.cpp.

```
82 {
83     return m_font;
84 }
```

7.50.3.4 void PrerenderedLabel::invalidate () const `[protected, virtual]`

invalidate() needs to be called if any of the properties that determine the visual appearance of the prerendered element change.

It can be called for a const object, as objects may need to force recalculation of the pixmap.

Implements PrerenderedElement.

Definition at line 70 of file KDChartTextLabelCache.cpp.

Referenced by setAngle(), setBrush(), setFont(), and setText().

```
71 {
72     m_dirty = true;
73 }
```

7.50.3.5 const QPen& PrerenderedLabel::pen () const

7.50.3.6 const QPixmap & PrerenderedLabel::pixmap () const `[virtual]`

Returns the rendered element.

If any of the properties have change, the element will be regenerated.

Implements PrerenderedElement.

Definition at line 119 of file KDChartTextLabelCache.cpp.

References INC_HIT_COUNT, and INC_MISS_COUNT.

```
120 {
121     if ( m_dirty ) {
122         INC_MISS_COUNT;
123         paint();
124     } else {
125         INC_HIT_COUNT;
126     }
127     return m_pixmap;
128 }
```

7.50.3.7 const QPointF & PrerenderedElement::position () const `[inherited]`

Get the position of the element.

Definition at line 40 of file KDChartTextLabelCache.cpp.

```
{
    return m_position;
}
```

7.50.3.8 KDChartEnums::PositionValue PrerenderedElement::referencePoint () const `[inherited]`

Get the reference point of the element.

Definition at line 50 of file KDChartTextLabelCache.cpp.

Referenced by referencePointLocation().

```
{
    return m_referencePoint;
}
```

7.50.3.9 QPointF PrerenderedLabel::referencePointLocation () const

Definition at line 233 of file KDChartTextLabelCache.cpp.

References PrerenderedElement::referencePoint().

```
{
    return referencePointLocation( referencePoint() );
}
```

7.50.3.10 QPointF PrerenderedLabel::referencePointLocation (KDChartEnums::PositionValue *position*) const `[virtual]`

Return the location of the reference point relatively to the pixmap's origin.

Implements PrerenderedElement.

Definition at line 238 of file KDChartTextLabelCache.cpp.

References INC_HIT_COUNT, and INC_MISS_COUNT.

```
{
    if ( m_dirty ) {
        INC_MISS_COUNT;
        paint();
    } else {
        INC_HIT_COUNT;
    }

    switch( position ) {
    case KDChartEnums::PositionCenter:
        return m_referenceBottomLeft + 0.5 * m_textBaseLineVector + 0.5 * m_textAscendVector;
    case KDChartEnums::PositionNorthWest:
        return m_referenceBottomLeft + m_textAscendVector;
    case KDChartEnums::PositionNorth:
        return m_referenceBottomLeft + 0.5 * m_textBaseLineVector + m_textAscendVector;
    case KDChartEnums::PositionNorthEast:
        return m_referenceBottomLeft + m_textBaseLineVector + m_textAscendVector;
    case KDChartEnums::PositionEast:
        return m_referenceBottomLeft + 0.5 * m_textAscendVector;
    case KDChartEnums::PositionSouthEast:
        return m_referenceBottomLeft + m_textBaseLineVector;
    case KDChartEnums::PositionSouth:
        return m_referenceBottomLeft + 0.5 * m_textBaseLineVector;
    case KDChartEnums::PositionSouthWest:
        return m_referenceBottomLeft;
    case KDChartEnums::PositionWest:
        return m_referenceBottomLeft + m_textBaseLineVector + 0.5 * m_textAscendVector;

    case KDChartEnums::PositionUnknown: // intentional fall-through
    case KDChartEnums::PositionFloating: // intentional fall-through
    default:
        return QPointF();
    }
}
```

7.50.3.11 void PrerenderedLabel::setAngle (double *angle*)

Definition at line 108 of file KDChartTextLabelCache.cpp.

References invalidate().

```
{
    m_angle = angle;
    invalidate();
}
```

7.50.3.12 void PrerenderedLabel::setBrush (const QBrush & *brush*)

Definition at line 97 of file KDChartTextLabelCache.cpp.

References invalidate().

```
{
    m_brush = brush;
    invalidate();
}
```

7.50.3.13 void PrerenderedLabel::setFont (const QFont & *font*)

Definition at line 75 of file KDChartTextLabelCache.cpp.

References invalidate().

```
{
    m_font = font;
    invalidate();
}
```

7.50.3.14 void PrerenderedLabel::setPen (const QPen &)

7.50.3.15 void PrerenderedElement::setPosition (const QPointF & *position*) `[inherited]`

Set the position of the element.

Definition at line 35 of file KDChartTextLabelCache.cpp.

```
{   // this does not invalidate the element
    m_position = position;
}
```

7.50.3.16 void PrerenderedElement::setReferencePoint (KDChartEnums::PositionValue) `[inherited]`

Set the reference point of the element.

Every element has nine possible reference points (all compass directions, plus the center.

Definition at line 45 of file KDChartTextLabelCache.cpp.

```
{   // this does not invalidate the element
    m_referencePoint = point;
}
```

7.50.3.17 void PrerenderedLabel::setText (const QString & *text*)

Definition at line 86 of file KDChartTextLabelCache.cpp.

References invalidate().

```
{
    m_text = text;
    invalidate();
}
```

7.50.3.18 const QString & PrerenderedLabel::text () const

Definition at line 92 of file KDChartTextLabelCache.cpp.

```
{
    return m_text;
}
```

The documentation for this class was generated from the following files:

- KDChartTextLabelCache.h
- KDChartTextLabelCache.cpp

7.51 QAbstractItemView Class Reference

Inheritance diagram for QAbstractItemView:

7.51.1 Detailed Description

Class only listed here to document inheritance of some KDChart classes.

Please consult the respective Qt documentation for details: `http://doc.trolltech.com/`

The documentation for this class was generated from the following file:

- KDChartChart.h

7.52 QAbstractProxyModel Class Reference

Inheritance diagram for QAbstractProxyModel:

7.52.1 Detailed Description

Class only listed here to document inheritance of some KDChart classes.

Please consult the respective Qt documentation for details: `http://doc.trolltech.com/`

The documentation for this class was generated from the following file:

- KDChartChart.h

7.53 QFrame Class Reference

Inheritance diagram for QFrame:

7.53.1 Detailed Description

Class only listed here to document inheritance of some KDChart classes.

Please consult the respective Qt documentation for details: `http://doc.trolltech.com/`

The documentation for this class was generated from the following file:

- KDChartChart.h

7.54 QLayoutItem Class Reference

Inheritance diagram for QLayoutItem:

The documentation for this class was generated from the following file:

- KDChartLayoutItems.h

7.55 QObject Class Reference

Inheritance diagram for QObject:

7.55.1 Detailed Description

Class only listed here to document inheritance of some KDChart classes.

Please consult the respective Qt documentation for details: `http://doc.trolltech.com/`

The documentation for this class was generated from the following file:

- KDChartChart.h

7.56 QSortFilterProxyModel Class Reference

Inheritance diagram for QSortFilterProxyModel:

7.56.1 Detailed Description

Class only listed here to document inheritance of some KDChart classes.

Please consult the respective Qt documentation for details: `http://doc.trolltech.com/`

The documentation for this class was generated from the following file:

- KDChartChart.h

7.57 QTextDocument Class Reference

Inheritance diagram for QTextDocument:

The documentation for this class was generated from the following file:

- KDTextDocument.h

7.58 QWidget Class Reference

Inheritance diagram for QWidget:

7.58.1 Detailed Description

Class only listed here to document inheritance of some KDChart classes.

Please consult the respective Qt documentation for details: `http://doc.trolltech.com/`

The documentation for this class was generated from the following file:

- KDChartChart.h

7.59 KDChart::RelativePosition Class Reference

`#include <KDChartRelativePosition.h>`

7.59.1 Detailed Description

Defines relative position information: reference area, position in this area, horizontal / vertical padding, and rotating.

Using RelativePosition you can specify the relative parts of some position information, and you can specify the absolute parts: the reference area, and the position in this area.

To get an absolute position, you have three options:

- either you declare both, the relative and the absolute parts, using setReferenceArea for the later,
- or you specify a set of points, using setReferencePoints,
- or you refrein from using either, but leave it to KD Chart to find a matching reference area for you.

Definition at line 62 of file KDChartRelativePosition.h.

Public Member Functions

- Qt::Alignment alignment () const
- const QPointF calculatedPoint (const QSizeF &autoSize) const

  Calculate a point, according to the reference area/position and horiz/vert padding.

- Measure horizontalPadding () const
- bool operator!= (const RelativePosition &other) const
- RelativePosition & operator= (const RelativePosition &other)
- bool operator== (const RelativePosition &) const
- QObject ∗ referenceArea () const
- const QPointF referencePoint () const

  Return the reference point, according to the reference area/position, but ignoring horiz/vert padding.

- const PositionPoints referencePoints () const
- Position referencePosition () const
- RelativePosition (const RelativePosition &)
- RelativePosition ()
- void resetReferencePosition ()

  Resets the position of the anchor point to the built-in default.

- qreal rotation () const
- void setAlignment (Qt::Alignment flags)

  Specifies the location of the content, that is to be positioned by this RelativePosition.

- void setHorizontalPadding (const Measure &padding)

  Specifies the horizontal width of the gap between the anchor point and the content, that is to be positioned by this RelativePosition.

- void setReferenceArea (QObject ∗area)

  Specifies the reference area to be used to find the anchor point.

- void setReferencePoints (const PositionPoints &points)

  Specifies a set of points from which the anchor point will be selected.

- void setReferencePosition (Position position)

  Specifies the position of the anchor point.

- void setRotation (qreal rot)
- void setVerticalPadding (const Measure &padding)

  Specifies the vertical width of the gap between the anchor point and the content, that is to be positioned by this RelativePosition.

- Measure verticalPadding () const
- ∼RelativePosition ()

7.59.2 Constructor & Destructor Documentation

7.59.2.1 KDChart::RelativePosition::RelativePosition ()

7.59.2.2 KDChart::RelativePosition::RelativePosition (const RelativePosition &)

7.59.2.3 KDChart::RelativePosition::∼RelativePosition ()

7.59.3 Member Function Documentation

7.59.3.1 Qt::Alignment KDChart::RelativePosition::alignment () const

Referenced by operator<<(), KDChart::AbstractDiagram::paintDataValueText(), and KDChart::Chart::reLayoutFloatingLegends().

7.59.3.2 const QPointF KDChart::RelativePosition::calculatedPoint (const QSizeF & *autoSize*) const

Calculate a point, according to the reference area/position and horiz/vert padding.

This method is called at drawing time: The returned point is used as anchor point. Note that calculatedPoint ignores the alignment setting, it just returns the point, so the calling code needs to take alignment into account explicitely.

See also:

referencePoint, setReferenceArea, setReferencePosition, setHorizontalPadding, setVerticalPadding

Referenced by KDChart::Chart::reLayoutFloatingLegends().

7.59.3.3 Measure KDChart::RelativePosition::horizontalPadding () const

Referenced by operator<<().

7.59.3.4 bool KDChart::RelativePosition::operator!= (const RelativePosition & *other*) const

Definition at line 198 of file KDChartRelativePosition.h.

References operator==().

```
198 { return !operator==( other ); }
```

7.59.3.5 RelativePosition& KDChart::RelativePosition::operator= (const RelativePosition & *other*)

7.59.3.6 bool KDChart::RelativePosition::operator== (const RelativePosition &) const

Referenced by operator!=().

7.59.3.7 QObject∗ KDChart::RelativePosition::referenceArea () const

Referenced by operator<<().

7.59.3.8 const QPointF KDChart::RelativePosition::referencePoint () const

Return the reference point, according to the reference area/position, but ignoring horiz/vert padding.

This method is called at drawing time. The returned point is used to test if the label of a data value is to be printed: labels are printed only, if their reference points are either inside or touching the coordinate plane.

See also:
calculatedPoint, setReferenceArea, setReferencePosition, setHorizontalPadding, setVerticalPadding

7.59.3.9 const PositionPoints KDChart::RelativePosition::referencePoints () const

7.59.3.10 Position KDChart::RelativePosition::referencePosition () const

Referenced by operator<<().

7.59.3.11 void KDChart::RelativePosition::resetReferencePosition ()

Resets the position of the anchor point to the built-in default.

If the anchor point of a RelativePosition is reset (or never changed from the default setting, resp.) KD Chart will choose an appropriate Position at run-time.

e.g. BarDiagrams will use Position::NorthWest / Position::SouthEast for positive / negative values.

See also:
setReferencePosition, setReferenceArea, setAlignment, setHorizontalPadding, setVerticalPadding, KDChart::Position

7.59.3.12 qreal KDChart::RelativePosition::rotation () const

Referenced by operator<<().

7.59.3.13 void KDChart::RelativePosition::setAlignment (Qt::Alignment *flags*)

Specifies the location of the content, that is to be positioned by this RelativePosition.

Aligning is applied, after horiz./vert. padding was retrieved to calculate the real reference point, so aligning is seen as relative to that point.

See also:
setReferencePosition, setReferenceArea, setHorizontalPadding, setVerticalPadding

7.59.3.14 void KDChart::RelativePosition::setHorizontalPadding (const Measure & *padding*)

Specifies the horizontal width of the gap between the anchor point and the content, that is to be positioned by this RelativePosition.

Note:
When printing data value texts the font height is used as reference size for both, horizontal and vertical padding, if the respective padding's Measure is using automatic reference area detection.

See also:
setVerticalPadding, setReferencePosition, setReferenceArea

7.59.3.15 void KDChart::RelativePosition::setReferenceArea (QObject ∗ *area*)

Specifies the reference area to be used to find the anchor point.

The reference area's type can be either QWidget, or be derived from KDChart::AbstractArea.

Note:
Usage of reference area and reference points works mutually exclusively: Only one setting can be valid, so any former specification of reference points is reset when you call setReferenceArea.

Also note: In a few cases KD Chart will ignore your area (or points, resp.) settings! Relative positioning of data value texts is an example: For these the reference area is the respective data area taking precendence over your settings.

See also:
setReferencePosition, setAlignment, setHorizontalPadding, setVerticalPadding

7.59.3.16 void KDChart::RelativePosition::setReferencePoints (const PositionPoints & *points*)

Specifies a set of points from which the anchor point will be selected.

Note:
Usage of reference area and reference points works mutually exclusively: Only one setting can be valid, so any former specification of reference area is reset when you call setReferencePoints.

Also note: In a few cases KD Chart will ignore your points (or area, resp.) settings! Relative positioning of data value texts is an example: For these the reference area is the respective data area taking precendence over your settings.

See also:

setReferenceArea, setReferencePosition, setAlignment, setHorizontalPadding, setVerticalPadding

7.59.3.17 void KDChart::RelativePosition::setReferencePosition (Position *position*)

Specifies the position of the anchor point.

The anchor point of a RelativePosition may be one of the pre-defined points of it's reference area - for details see KDChart::Position.

See also:

resetReferencePosition, setReferenceArea, setAlignment, setHorizontalPadding, setVerticalPadding, KDChart::Position

7.59.3.18 void KDChart::RelativePosition::setRotation (qreal *rot*)

7.59.3.19 void KDChart::RelativePosition::setVerticalPadding (const Measure & *padding*)

Specifies the vertical width of the gap between the anchor point and the content, that is to be positioned by this RelativePosition.

Note:

When printing data value texts the font height is used as reference size for both, horizontal and vertical padding, if the respective padding's Measure is using automatic reference area detection.

See also:

setHorizontalPadding, setReferencePosition, setReferenceArea

7.59.3.20 Measure KDChart::RelativePosition::verticalPadding () const

Referenced by operator<<().

The documentation for this class was generated from the following file:

- KDChartRelativePosition.h

7.60 KDChart::RingDiagram Class Reference

`#include <KDChartRingDiagram.h>`

Inheritance diagram for KDChart::RingDiagram:Collaboration diagram for KDChart::RingDiagram:

Public Member Functions

- bool allowOverlappingDataValueTexts () const
- bool antiAliasing () const
- virtual AttributesModel ∗ attributesModel () const

  Returns the AttributesModel, that is used by this diagram.

- QBrush brush (const QModelIndex &index) const

  Retrieve the brush to be used, for painting the datapoint at the given index in the model.

- QBrush brush (int dataset) const

  Retrieve the brush to be used for the given dataset.

- QBrush brush () const

  Retrieve the brush to be used for painting datapoints globally.

- virtual RingDiagram ∗ clone () const
- int columnCount () const
- bool compare (const AbstractDiagram ∗other) const

  Returns true if both diagrams have the same settings.

- AbstractCoordinatePlane ∗ coordinatePlane () const

  The coordinate plane associated with the diagram.

- const QPair< QPointF, QPointF > dataBoundaries () const

  Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

- virtual void dataChanged (const QModelIndex &topLeft, const QModelIndex &bottomRight)

  [reimplemented]

- QList< QBrush > datasetBrushes () const

  The set of dataset brushes currently used, for use in legends, etc.

- int datasetDimension () const

  The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

- QStringList datasetLabels () const

  The set of dataset labels currently displayed, for use in legends, etc.

- QList< MarkerAttributes > datasetMarkers () const

  The set of dataset markers currently used, for use in legends, etc.

- QList< QPen > datasetPens () const

  The set of dataset pens currently used, for use in legends, etc.

- DataValueAttributes dataValueAttributes (const QModelIndex &index) const

  Retrieve the DataValueAttributes for the given index.

- DataValueAttributes dataValueAttributes (int column) const

  Retrieve the DataValueAttributes for the given dataset.

- DataValueAttributes dataValueAttributes () const

  Retrieve the DataValueAttributes speficied globally.

- virtual void doItemsLayout ()

  [reimplemented]

- qreal granularity () const
- virtual int horizontalOffset () const

  [reimplemented]

- virtual QModelIndex indexAt (const QPoint &point) const

  [reimplemented]

- bool isHidden (const QModelIndex &index) const

  Retrieve the hidden status for the given index.

- bool isHidden (int column) const

  Retrieve the hidden status for the given dataset.

- bool isHidden () const

  Retrieve the hidden status speficied globally.

- virtual bool isIndexHidden (const QModelIndex &index) const

  [reimplemented]

- QStringList itemRowLabels () const

  The set of item row labels currently displayed, for use in Abscissa axes, etc.

- virtual QModelIndex moveCursor (CursorAction cursorAction, Qt::KeyboardModifiers modifiers)

  [reimplemented]

- virtual double numberOfGridRings () const

  [reimplemented]

- virtual double numberOfValuesPerDataset () const

  [reimplemented]

- void paintDataValueText (QPainter ∗painter, const QModelIndex &index, const QPointF &pos, double value)
- void paintMarker (QPainter ∗painter, const QModelIndex &index, const QPointF &pos)

- virtual void paintMarker (QPainter ∗painter, const MarkerAttributes &markerAttributes, const QBrush &brush, const QPen &, const QPointF &point, const QSizeF &size)
- QPen pen (const QModelIndex &index) const

  Retrieve the pen to be used, for painting the datapoint at the given index in the model.

- QPen pen (int dataset) const

  Retrieve the pen to be used for the given dataset.

- QPen pen () const

  Retrieve the pen to be used for painting datapoints globally.

- bool percentMode () const
- PieAttributes pieAttributes (const QModelIndex &index) const
- PieAttributes pieAttributes (int column) const
- PieAttributes pieAttributes () const
- const PolarCoordinatePlane ∗ polarCoordinatePlane () const
- bool relativeThickness () const
- virtual void resize (const QSizeF &area)

  [reimplemented]

- RingDiagram (QWidget ∗parent=0, PolarCoordinatePlane ∗plane=0)
- virtual void scrollTo (const QModelIndex &index, ScrollHint hint=EnsureVisible)

  [reimplemented]

- void setAllowOverlappingDataValueTexts (bool allow)

  Set whether data value labels are allowed to overlap.

- void setAntiAliasing (bool enabled)

  Set whether anti-aliasing is to be used while rendering this diagram.

- virtual void setAttributesModel (AttributesModel ∗model)

  Associate an AttributesModel with this diagram.

- void setBrush (const QBrush &brush)

  Set the brush to be used, for painting all datasets in the model.

- void setBrush (int dataset, const QBrush &brush)

  Set the brush to be used, for painting the given dataset.

- void setBrush (const QModelIndex &index, const QBrush &brush)

  Set the brush to be used, for painting the datapoint at the given index.

- virtual void setCoordinatePlane (AbstractCoordinatePlane ∗plane)

  Set the coordinate plane associated with the diagram.

- void setDatasetDimension (int dimension)

  Sets the dataset dimension of the diagram.

- void setDataValueAttributes (const DataValueAttributes &a)

  Set the DataValueAttributes for all datapoints in the model.

- void setDataValueAttributes (int dataset, const DataValueAttributes &a)

  Set the DataValueAttributes for the given dataset.

- void setDataValueAttributes (const QModelIndex &index, const DataValueAttributes &a)

  Set the DataValueAttributes for the given index.

- void setGranularity (qreal value)

  Set the granularity: the smaller the granularity the more your diagram segments will show facettes instead of rounded segments.

- void setHidden (bool hidden)

  Hide (or unhide, resp.) all datapoints in the model.

- void setHidden (int column, bool hidden)

  Hide (or unhide, resp.) a dataset.

- void setHidden (const QModelIndex &index, bool hidden)

  Hide (or unhide, resp.) a data cell.

- virtual void setModel (QAbstractItemModel ∗model)

  Associate a model with the diagram.

- void setPen (const QPen &pen)

  Set the pen to be used, for painting all datasets in the model.

- void setPen (int dataset, const QPen &pen)

  Set the pen to be used, for painting the given dataset.

- void setPen (const QModelIndex &index, const QPen &pen)

  Set the pen to be used, for painting the datapoint at the given index.

- void setPercentMode (bool percent)
- void setPieAttributes (int column, const PieAttributes &a)
- void setPieAttributes (const PieAttributes &a)
- void setRelativeThickness (bool relativeThickness)
- virtual void setRootIndex (const QModelIndex &idx)

  Set the root index in the model, where the diagram starts referencing data for display.

- virtual void setSelection (const QRect &rect, QItemSelectionModel::SelectionFlags command)

  [reimplemented]

- void setStartPosition (int degrees)
- void setThreeDPieAttributes (const QModelIndex &index, const ThreeDPieAttributes &a)
- void setThreeDPieAttributes (int column, const ThreeDPieAttributes &a)
- void setThreeDPieAttributes (const ThreeDPieAttributes &a)
- int startPosition () const
- ThreeDPieAttributes threeDPieAttributes (const QModelIndex &index) const
- ThreeDPieAttributes threeDPieAttributes (int column) const
- ThreeDPieAttributes threeDPieAttributes () const

- void update () const
- void useDefaultColors ()

  Set the palette to be used, for painting datasets to the default palette.

- void useRainbowColors ()

  Set the palette to be used, for painting datasets to the rainbow palette.

- virtual bool usesExternalAttributesModel () const

  Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

- void useSubduedColors ()

  Set the palette to be used, for painting datasets to the subdued palette.

- virtual double valueTotals () const

  [reimplemented]

- virtual int verticalOffset () const

  [reimplemented]

- virtual QRect visualRect (const QModelIndex &index) const

  [reimplemented]

- virtual QRegion visualRegionForSelection (const QItemSelection &selection) const

  [reimplemented]

- virtual ∼RingDiagram ()

Protected Member Functions

- QModelIndex attributesModelRootIndex () const
- virtual const QPair< QPointF, QPointF > calculateDataBoundaries () const

  [reimplemented]

- virtual bool checkInvariants (bool justReturnTheStatus=false) const
- QModelIndex columnToIndex (int column) const
- void dataHidden ()

  This signal is emitted, when the hidden status of at least one data cell was (un)set.

- void modelsChanged ()

  This signal is emitted, when either the model or the AttributesModel is replaced.

- virtual void paint (PaintContext ∗paintContext)

  [reimplemented]

- virtual void paintDataValueTexts (QPainter ∗painter)
- void paintEvent (QPaintEvent ∗)
- virtual void paintMarkers (QPainter ∗painter)
- void propertiesChanged ()

Emitted upon change of a property of the Diagram.

- void resizeEvent (QResizeEvent ∗)
- void setAttributesModelRootIndex (const QModelIndex &)
- void setDataBoundariesDirty () const
- double valueForCell (int row, int column) const

  Helper method, retrieving the data value (DisplayRole) for a given row and column.

Protected Attributes

- Q_SIGNALS __pad0__: void layoutChanged(AbstractDiagram∗)

7.60.1 Constructor & Destructor Documentation

7.60.1.1 RingDiagram::RingDiagram (QWidget ∗ *parent* = 0, PolarCoordinatePlane ∗ *plane* = 0) `[explicit]`

Definition at line 50 of file KDChartRingDiagram.cpp.

Referenced by clone().

```
50                                                                            :
51      AbstractPieDiagram( new Private(), parent, plane )
52 {
53      init();
54 }
```

7.60.1.2 RingDiagram::∼RingDiagram () `[virtual]`

Definition at line 56 of file KDChartRingDiagram.cpp.

```
57 {
58 }
```

7.60.2 Member Function Documentation

7.60.2.1 bool AbstractDiagram::allowOverlappingDataValueTexts () const `[inherited]`

Returns:

Whether data value labels are allowed to overlap.

Definition at line 446 of file KDChartAbstractDiagram.cpp.

References d.

```
450 {
```

7.60.2.2 bool AbstractDiagram::antiAliasing () const `[inherited]`

Returns:

Whether anti-aliasing is to be used for rendering this diagram.

Definition at line 457 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::paint().

```
461 {
```

7.60.2.3 AttributesModel ∗ AbstractDiagram::attributesModel () const `[virtual, inherited]`

Returns the AttributesModel, that is used by this diagram.

By default each diagram owns its own AttributesModel, which should never be deleted. Only if a user-supplied AttributesModel has been set does the pointer returned here not belong to the diagram.

Returns:

The AttributesModel associated with the diagram.

See also:

setAttributesModel

Definition at line 286 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by paint(), KDChart::PolarDiagram::paint(), and KDChart::BarDiagram::setBarAttributes().

```
287 {
288     return d->attributesModel;
289 }
```

7.60.2.4 QModelIndex AbstractDiagram::attributesModelRootIndex () const `[protected, inherited]`

returns a QModelIndex pointing into the AttributesModel that corresponds to the root index of the diagram.

Definition at line 310 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::calculateDataBoundaries(), KDChart::BarDiagram::calculateDataBoundaries(), KDChart::LineDiagram::numberOfAbscissaSegments(), KDChart::BarDiagram::numberOfAbscissaSegments(), KDChart::LineDiagram::numberOfOrdinateSegments(), KDChart::BarDiagram::numberOfOrdinateSegments(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and KDChart::AbstractDiagram::valueForCell().

```
316 {
```

7.60.2.5 QBrush AbstractDiagram::brush (const QModelIndex & *index*) const `[inherited]`

Retrieve the brush to be used, for painting the datapoint at the given index in the model.

Parameters:

index The index of the datapoint in the model.

Returns:

The brush to use for painting.

Definition at line 816 of file KDChartAbstractDiagram.cpp.

```
822                                 :
QRect AbstractDiagram::visualRect(const QModelIndex &) const
```

7.60.2.6 QBrush AbstractDiagram::brush (int *dataset*) const `[inherited]`

Retrieve the brush to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the brush for.

Returns:

The brush to use for painting.

Definition at line 808 of file KDChartAbstractDiagram.cpp.

```
815 {
```

7.60.2.7 QBrush AbstractDiagram::brush () const `[inherited]`

Retrieve the brush to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The brush to use for painting.

Definition at line 802 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), and KDChart::AbstractDiagram::paintMarker().

```
807 {
```

7.60.2.8 const QPair< QPointF, QPointF > RingDiagram::calculateDataBoundaries () const `[protected, virtual]`

[reimplemented]

Implements KDChart::AbstractDiagram.

Definition at line 79 of file KDChartRingDiagram.cpp.

References KDChart::AbstractDiagram::checkInvariants().

```
80 {
81     if ( !checkInvariants(true) ) return QPair<QPointF, QPointF>( QPointF( 0, 0 ), QPointF( 0, 0 ) );
82
83     QPointF bottomLeft ( QPointF( 0, 0 ) );
84     QPointF topRight ( QPointF( 1, 1 ) );
85     return QPair<QPointF, QPointF> ( bottomLeft,  topRight );
86 }
```

7.60.2.9 bool AbstractDiagram::checkInvariants (bool *justReturnTheStatus* = false) const `[protected, virtual, inherited]`

Definition at line 930 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane().

Referenced by calculateDataBoundaries(), KDChart::PolarDiagram::calculateDataBoundaries(), KDChart::PieDiagram::calculateDataBoundaries(), KDChart::LineDiagram::calculateDataBoundaries(), KDChart::BarDiagram::calculateDataBoundaries(), paint(), KDChart::PolarDiagram::paint(), KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and KDChart::AbstractDiagram::paintMarker().

```
930                                 {
931         Q_ASSERT_X ( model(), "AbstractDiagram::checkInvariants()",
932                     "There is no usable model set, for the diagram." );
933
934         Q_ASSERT_X ( coordinatePlane(), "AbstractDiagram::checkInvariants()",
935                     "There is no usable coordinate plane set, for the diagram." );
936     }
937     return model() && coordinatePlane();
938 }
939
940 int AbstractDiagram::datasetDimension( ) const
```

7.60.2.10 RingDiagram * RingDiagram::clone () const `[virtual]`

Definition at line 64 of file KDChartRingDiagram.cpp.

References d, and RingDiagram().

```
65 {
66     return new RingDiagram( new Private( *d ) );
67 }
```

7.60.2.11 int AbstractPolarDiagram::columnCount () const `[inherited]`

Definition at line 60 of file KDChartAbstractPolarDiagram.cpp.

References KDChart::AbstractPolarDiagram::numberOfValuesPerDataset().

Referenced by KDChart::PieDiagram::calculateDataBoundaries(), KDChart::PieDiagram::paint(), and KDChart::PieDiagram::valueTotals().

```
61 {
62     return static_cast<int>( numberOfValuesPerDataset() );
63 }
```

7.60.2.12 QModelIndex AbstractDiagram::columnToIndex (int *column*) const `[protected, inherited]`

Definition at line 317 of file KDChartAbstractDiagram.cpp.

```
323 {
```

7.60.2.13 bool AbstractDiagram::compare (const AbstractDiagram ∗ *other*) const `[inherited]`

Returns true if both diagrams have the same settings.

Definition at line 135 of file KDChartAbstractDiagram.cpp.

```
136 {
137     if( other == this ) return true;
138     if( ! other ){
139         //qDebug() << "AbstractDiagram::compare() cannot compare to Null pointer";
140         return false;
141     }
142     /*
143     qDebug() << "\n             AbstractDiagram::compare() QAbstractScrollArea:";
144             // compare QAbstractScrollArea properties
145     qDebug() <<
146             ((horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
147             (verticalScrollBarPolicy()    == other->verticalScrollBarPolicy()));
148     qDebug() << "AbstractDiagram::compare() QFrame:";
149             // compare QFrame properties
150     qDebug() <<
151             ((frameShadow() == other->frameShadow()) &&
152             (frameShape()   == other->frameShape()) &&
153             (frameWidth()   == other->frameWidth()) &&
154             (lineWidth()    == other->lineWidth()) &&
155             (midLineWidth() == other->midLineWidth()));
156     qDebug() << "AbstractDiagram::compare() QAbstractItemView:";
157             // compare QAbstractItemView properties
158     qDebug() <<
159             ((alternatingRowColors() == other->alternatingRowColors()) &&
160             (hasAutoScroll()         == other->hasAutoScroll()) &&
161 #if QT_VERSION > 0x040199
162             (dragDropMode()          == other->dragDropMode()) &&
163             (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
164             (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
165             (verticalScrollMode()    == other->verticalScrollMode()) &&
166 #endif
167             (dragEnabled()           == other->dragEnabled()) &&
            (editTriggers()          == other->editTriggers()) &&
            (iconSize()              == other->iconSize()) &&
            (selectionBehavior()     == other->selectionBehavior()) &&
            (selectionMode()         == other->selectionMode()) &&
            (showDropIndicator()     == other->showDropIndicator()) &&
            (tabKeyNavigation()      == other->tabKeyNavigation()) &&
            (textElideMode()         == other->textElideMode()));
    qDebug() << "AbstractDiagram::compare() AttributesModel: ";
            // compare all of the properties stored in the attributes model
    qDebug() << attributesModel()->compare( other->attributesModel() );
    qDebug() << "AbstractDiagram::compare() own:";
            // compare own properties
    qDebug() <<
            ((rootIndex().column()            == other->rootIndex().column()) &&
            (rootIndex().row()                == other->rootIndex().row()) &&
            (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
            (antiAliasing()                   == other->antiAliasing()) &&
            (percentMode()                    == other->percentMode()) &&
            (datasetDimension()               == other->datasetDimension()));
    */
    return  // compare QAbstractScrollArea properties
            (horizontalScrollBarPolicy() == other->horizontalScrollBarPolicy()) &&
            (verticalScrollBarPolicy()   == other->verticalScrollBarPolicy()) &&
            // compare QFrame properties
            (frameShadow()   == other->frameShadow()) &&
            (frameShape()    == other->frameShape()) &&
            (frameWidth()    == other->frameWidth()) &&
            (lineWidth()     == other->lineWidth()) &&
            (midLineWidth() == other->midLineWidth()) &&
            // compare QAbstractItemView properties
            (alternatingRowColors()  == other->alternatingRowColors()) &&
            (hasAutoScroll()         == other->hasAutoScroll()) &&
#if QT_VERSION > 0x040199
            (dragDropMode()          == other->dragDropMode()) &&
            (dragDropOverwriteMode() == other->dragDropOverwriteMode()) &&
            (horizontalScrollMode()  == other->horizontalScrollMode ()) &&
            (verticalScrollMode()    == other->verticalScrollMode()) &&
#endif
            (dragEnabled()           == other->dragEnabled()) &&
            (editTriggers()          == other->editTriggers()) &&
            (iconSize()              == other->iconSize()) &&
            (selectionBehavior()     == other->selectionBehavior()) &&
            (selectionMode()         == other->selectionMode()) &&
            (showDropIndicator()     == other->showDropIndicator()) &&
            (tabKeyNavigation()      == other->tabKeyNavigation()) &&
            (textElideMode()         == other->textElideMode()) &&
            // compare all of the properties stored in the attributes model
            attributesModel()->compare( other->attributesModel() ) &&
            // compare own properties
            (rootIndex().column()             == other->rootIndex().column()) &&
            (rootIndex().row()                == other->rootIndex().row()) &&
            (allowOverlappingDataValueTexts() == other->allowOverlappingDataValueTexts()) &&
            (antiAliasing()                   == other->antiAliasing()) &&
            (percentMode()                    == other->percentMode()) &&
            (datasetDimension()               == other->datasetDimension());
}
```

7.60.2.14 AbstractCoordinatePlane ∗ AbstractDiagram::coordinatePlane () const `[inherited]`

The coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. By default this is a CartesianCoordinatePlane.

Returns:

The coordinate plane associated with the diagram.

Definition at line 226 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractCartesianDiagram::layoutPlanes(), KDChart::PolarDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), KDChart::AbstractPolarDiagram::polarCoordinatePlane(), and KDChart::AbstractCartesianDiagram::setCoordinatePlane().

```
227 {
228     return d->plane;
229 }
```

7.60.2.15 const QPair< QPointF, QPointF > AbstractDiagram::dataBoundaries () const `[inherited]`

Return the bottom left and top right data point, that the diagram will display (unless the grid adjusts these values).

This method returns a chached result of calculations done by calculateDataBoundaries. Classes derived from AbstractDiagram must implement the calculateDataBoundaries function, to specify their own way of calculating the data boundaries. If derived classes want to force recalculation of the data boundaries, they can call setDataBoundariesDirty()

Returned value is in diagram coordinates.

Definition at line 231 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::calculateDataBoundaries(), and d.

Referenced by KDChart::CartesianCoordinatePlane::getRawDataBoundingRectFromDiagrams(), KDChart::PolarCoordinatePlane::layoutDiagrams(), KDChart::LineDiagram::paint(), and KDChart::BarDiagram::paint().

```
232 {
233     if( d->databoundariesDirty ){
234         d->databoundaries = calculateDataBoundaries ();
235         d->databoundariesDirty = false;
236     }
237     return d->databoundaries;
238 }
```

7.60.2.16 void AbstractDiagram::dataChanged (const QModelIndex & *topLeft*, const QModelIndex & *bottomRight*) `[virtual, inherited]`

[reimplemented]

Definition at line 338 of file KDChartAbstractDiagram.cpp.

References d.

```
338 {
339   // We are still too dumb to do intelligent updates...
340   d->databoundariesDirty = true;
341   scheduleDelayedItemsLayout();
342 }
343
344
```

7.60.2.17 void KDChart::AbstractDiagram::dataHidden () [protected, inherited]

This signal is emitted, when the hidden status of at least one data cell was (un)set.

7.60.2.18 QList< QBrush > AbstractDiagram::datasetBrushes () const [inherited]

The set of dataset brushes currently used, for use in legends, etc.

Note:

Cell-level override brushes, if set, take precedence over the dataset values, so you might need to check these too, in order to find the brush, that is used for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 894 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), KDChart::Legend::datasetCount(), and KDChart::Legend::setBrushesFromDiagram().

```
896                                                                                  {
897         QBrush brush = qVariantValue<QBrush>( attributesModel()->headerData( i, Qt::Vertical, DatasetE
898         ret << brush;
899     }
900
901     return ret;
902 }
903
904 QList<QPen> AbstractDiagram::datasetPens() const
```

7.60.2.19 int AbstractDiagram::datasetDimension () const [inherited]

The dataset dimension of a diagram determines, how many value dimensions it expects each datapoint to have.

For each dimension it will expect one column of values in the model. If the dimensionality is 1, automatic values will be used for the abscissa.

For example a diagram with the default dimension of 1, will have one column per datapoint (the y values) and will use automatic values for the x axis (1, 2, 3, ... n). If the dimension is 2, the diagram will use the first, (and the third, fifth, etc) columns as X values, and the second, (and the fourth, sixth, etc) column as Y values.

Returns:

The dataset dimension of the diagram.

Definition at line 942 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::calculateDataBoundaries(), KDChart::LineDiagram::getCellValues(), KDChart::CartesianCoordinatePlane::getDataDimensionsList(), KDChart::LineDiagram::paint(), and KDChart::LineDiagram::setType().

```
946 {
```

7.60.2.20 QStringList AbstractDiagram::datasetLabels () const `[inherited]`

The set of dataset labels currently displayed, for use in legends, etc.

Returns:

The set of dataset labels currently displayed.

Definition at line 882 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), and KDChart::Legend::datasetCount().

```
883                                                  : " << attributesModel()->columnCount(attributesModel
884     const int columnCount = attributesModel()->columnCount(attributesModelRootIndex());
885     for( int i = datasetDimension()-1; i < columnCount; i += datasetDimension() ){
886         //qDebug() << "dataset label: " << attributesModel()->headerData( i, Qt::Horizontal, Qt::Displ
887         ret << attributesModel()->headerData( i, Qt::Horizontal, Qt::DisplayRole ).toString();
888     }
889     return ret;
890 }
891
892 QList<QBrush> AbstractDiagram::datasetBrushes() const
```

7.60.2.21 QList< MarkerAttributes > AbstractDiagram::datasetMarkers () const `[inherited]`

The set of dataset markers currently used, for use in legends, etc.

Note:

Cell-level override markers, if set, take precedence over the dataset values, so you might need to check these too, in order to find the marker, that is shown for a single cell.

Returns:

The current set of dataset brushes.

Definition at line 917 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
919                                                                                   {
920         DataValueAttributes a =
921             qVariantValue<DataValueAttributes>( attributesModel()->headerData( i, Qt::Vertical, DataVa
922         const MarkerAttributes &ma = a.markerAttributes();
923         ret << ma;
924     }
925     return ret;
926 }
927
928 bool AbstractDiagram::checkInvariants( bool justReturnTheStatus ) const
```

7.60.2.22 QList< QPen > AbstractDiagram::datasetPens () const `[inherited]`

The set of dataset pens currently used, for use in legends, etc.

Note:

Cell-level override pens, if set, take precedence over the dataset values, so you might need to check these too, in order to find the pens, that is used for a single cell.

Returns:

The current set of dataset pens.

Definition at line 906 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend().

```
908                                                                                    {
909         QPen pen = qVariantValue<QPen>( attributesModel()->headerData( i, Qt::Vertical, DatasetPenRole
910         ret << pen;
911     }
912     return ret;
913 }
914
915 QList<MarkerAttributes> AbstractDiagram::datasetMarkers() const
```

7.60.2.23 DataValueAttributes AbstractDiagram::dataValueAttributes (const QModelIndex & *index*) const `[inherited]`

Retrieve the DataValueAttributes for the given index.

This will fall back automatically to what was set at dataset or model level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the attributes for.

Returns:

The DataValueAttributes for the given index.

Definition at line 427 of file KDChartAbstractDiagram.cpp.

```
433 {
```

7.60.2.24 DataValueAttributes AbstractDiagram::dataValueAttributes (int *column*) const `[inherited]`

Retrieve the DataValueAttributes for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the attributes for.

Returns:

The DataValueAttributes for the given dataset.

Definition at line 420 of file KDChartAbstractDiagram.cpp.

```
426 {
```

7.60.2.25 DataValueAttributes AbstractDiagram::dataValueAttributes () const `[inherited]`

Retrieve the DataValueAttributes speficied globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:
The global DataValueAttributes.

Definition at line 414 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::AbstractDiagram::paintDataValueText(), and KDChart::AbstractDiagram::paintMarker().

```
419 {
```

7.60.2.26 void AbstractDiagram::doItemsLayout () `[virtual, inherited]`

[reimplemented]

Definition at line 329 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::AbstractDiagram::update().

```
329                     {
330         d->plane->layoutDiagrams();
331         update();
332     }
333     QAbstractItemView::doItemsLayout();
334 }
335
336 void AbstractDiagram::dataChanged( const QModelIndex &topLeft,
```

7.60.2.27 qreal AbstractPieDiagram::granularity () const `[inherited]`

Returns:
the granularity.

Definition at line 69 of file KDChartAbstractPieDiagram.cpp.

References d.

Referenced by KDChart::PieDiagram::paint().

```
70 {
71     return (d->granularity < 0.05 || d->granularity > 36.0)
72             ? 1.0
73     : d->granularity;
74 }
```

7.60.2.28 int AbstractDiagram::horizontalOffset () const `[virtual, inherited]`

[reimplemented]

Definition at line 839 of file KDChartAbstractDiagram.cpp.

```
841 { return 0; }
```

7.60.2.29 QModelIndex AbstractDiagram::indexAt (const QPoint & *point*) const `[virtual, inherited]`

[reimplemented]

Definition at line 833 of file KDChartAbstractDiagram.cpp.

```
835 { return QModelIndex(); }
```

7.60.2.30 bool AbstractDiagram::isHidden (const QModelIndex & *index*) const `[inherited]`

Retrieve the hidden status for the given index.

This will fall back automatically to what was set at dataset or diagram level, if there are no datapoint specific settings.

Parameters:

index The datapoint to retrieve the hidden status for.

Returns:

The hidden status for the given index.

Definition at line 386 of file KDChartAbstractDiagram.cpp.

7.60.2.31 bool AbstractDiagram::isHidden (int *column*) const `[inherited]`

Retrieve the hidden status for the given dataset.

This will fall back automatically to what was set at diagram level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the hidden status for.

Returns:

The hidden status for the given dataset.

Definition at line 379 of file KDChartAbstractDiagram.cpp.

```
385 {
```

7.60.2.32 bool AbstractDiagram::isHidden () const `[inherited]`

Retrieve the hidden status speficied globally.

This will fall back automatically to the default settings (= not hidden), if there are no specific settings.

Returns:

The global hidden status.

Definition at line 373 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::Legend::buildLegend(), KDChart::LineDiagram::paint(), and KDChart::LineDiagram::valueForCellTesting().

```
{
```

7.60.2.33 bool AbstractDiagram::isIndexHidden (const QModelIndex & *index*) const `[virtual, inherited]`

[reimplemented]

Definition at line 845 of file KDChartAbstractDiagram.cpp.

```
{}
```

7.60.2.34 QStringList AbstractDiagram::itemRowLabels () const `[inherited]`

The set of item row labels currently displayed, for use in Abscissa axes, etc.

Returns:

The set of item row labels currently displayed.

Definition at line 870 of file KDChartAbstractDiagram.cpp.

```
                                                 : " << attributesModel()->rowCount(attributesModelRoo
    const int rowCount = attributesModel()->rowCount(attributesModelRootIndex());
    for( int i = 0; i < rowCount; ++i ){
        //qDebug() << "item row label: " << attributesModel()->headerData( i, Qt::Vertical, Qt::Displa
        ret << attributesModel()->headerData( i, Qt::Vertical, Qt::DisplayRole ).toString();
    }
    return ret;
}

QStringList AbstractDiagram::datasetLabels() const
```

7.60.2.35 void KDChart::AbstractDiagram::modelsChanged () `[protected, inherited]`

This signal is emitted, when either the model or the AttributesModel is replaced.

Referenced by KDChart::AbstractDiagram::setAttributesModel(), and KDChart::AbstractDiagram::setModel().

7.60.2.36 QModelIndex AbstractDiagram::moveCursor (CursorAction *cursorAction*, Qt::KeyboardModifiers *modifiers*) `[virtual, inherited]`

[reimplemented]

Definition at line 836 of file KDChartAbstractDiagram.cpp.

```
838 { return 0; }
```

7.60.2.37 double RingDiagram::numberOfGridRings () const `[virtual]`

[reimplemented]

Implements KDChart::AbstractPolarDiagram.

Definition at line 150 of file KDChartRingDiagram.cpp.

```
151 {
152     return 1;
153 }
```

7.60.2.38 double RingDiagram::numberOfValuesPerDataset () const `[virtual]`

[reimplemented]

Implements KDChart::AbstractPolarDiagram.

Definition at line 144 of file KDChartRingDiagram.cpp.

```
145 {
146     return model() ? model()->columnCount(rootIndex()) : 0.0;
147 }
```

7.60.2.39 void RingDiagram::paint (PaintContext ∗ *paintContext*) `[protected, virtual]`

[reimplemented]

Implements KDChart::AbstractDiagram.

Definition at line 101 of file KDChartRingDiagram.cpp.

References KDChart::AbstractDiagram::attributesModel(), KDChart::AbstractDiagram::checkInvariants(), KDChart::AttributesModel::headerData(), KDChart::AbstractDiagram::paintDataValueText(), and KDChart::PaintContext::painter().

Referenced by paintEvent().

```
102 {
103     // note: Not having any data model assigned is no bug
104     //       but we can not draw a diagram then either.
105     if ( !checkInvariants(true) )
106         return;
107
108     const int colCount = model()->columnCount(rootIndex());
109     DataValueTextInfoList list;
110     for ( int j=0; j<colCount; ++j ) {
        QBrush brush = qVariantValue<QBrush>( attributesModel()->headerData( j, Qt::Vertical, KDChart:
        PainterSaver painterSaver( ctx->painter() );
        ctx->painter()->setRenderHint ( QPainter::Antialiasing );
        ctx->painter()->setBrush( brush );
        QPen p( ctx->painter()->pen() );
        p.setColor( brush.color() );
        p.setWidth( 2 );// FIXME properties, use DatasetPenRole
        ctx->painter()->setPen( p );
        //ctx->painter()->drawPolyline( polygon );
    }
    DataValueTextInfoListIterator it( list );
    while ( it.hasNext() ) {
        const DataValueTextInfo& info = it.next();
        paintDataValueText( ctx->painter(), info.index, info.pos, info.value );
    }
}
```

7.60.2.40 void AbstractDiagram::paintDataValueText (QPainter ∗ *painter*, const QModelIndex & *index*, const QPointF & *pos*, double *value*) `[inherited]`

Definition at line 474 of file KDChartAbstractDiagram.cpp.

References KDChart::RelativePosition::alignment(), KDChart::TextAttributes::calculatedFont(), d, KDChart::DataValueAttributes::dataLabel(), KDChart::AbstractDiagram::dataValueAttributes(), KDChart::DataValueAttributes::decimalDigits(), KDChart::TextAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::TextAttributes::pen(), KDChart::DataValueAttributes::position(), KDChart::DataValueAttributes::prefix(), KDChart::TextAttributes::rotation(), KDChart::DataValueAttributes::showRepetitiveDataLabels(), KDChart::DataValueAttributes::suffix(), and KDChart::DataValueAttributes::textAttributes().

Referenced by paint(), and KDChart::PolarDiagram::paint().

```
{
    // paint one data series
    const DataValueAttributes a( dataValueAttributes(index) );
    if ( !a.isVisible() ) return;

    // handle decimal digits
    int decimalDigits = a.decimalDigits();
    int decimalPos = QString::number(  value ).indexOf( QLatin1Char( '.' ) );
    QString roundedValue;
    if ( a.dataLabel().isNull() ) {
        if ( decimalPos > 0 && value != 0 )
            roundedValue =  roundValues ( value, decimalPos, decimalDigits );
        else
            roundedValue = QString::number(  value );
    } else
        roundedValue = a.dataLabel();
        // handle prefix and suffix
    if ( !a.prefix().isNull() )
        roundedValue.prepend( a.prefix() );

    if ( !a.suffix().isNull() )
        roundedValue.append( a.suffix() );

    const TextAttributes ta( a.textAttributes() );
    // FIXME draw the non-text bits, background, etc
    if ( ta.isVisible() ) {

        QPointF pt( pos );
        /* for debugging:
        PainterSaver painterSaver( painter );
        painter->setPen( Qt::black );
        painter->drawLine( pos - QPointF( 1,1), pos + QPointF( 1,1) );
        painter->drawLine( pos - QPointF(-1,1), pos + QPointF(-1,1) );
        */

        // adjust the text start point position, if alignment is not Bottom/Left
        const RelativePosition relPos( a.position( value >= 0.0 ) );
        const Qt::Alignment alignBottomLeft = Qt::AlignBottom | Qt::AlignLeft;
        const QFont calculatedFont( ta.calculatedFont( d->plane, KDChartEnums::MeasureOrientationMinim
        //qDebug() << "calculatedFont's point size:" << calculatedFont.pointSizeF();
        if( (relPos.alignment() & alignBottomLeft) != alignBottomLeft ){
            const QRectF boundRect(
                    d->cachedFontMetrics( calculatedFont, this )->boundingRect( roundedValue ) );
            if( relPos.alignment() & Qt::AlignRight )
                pt.rx() -= boundRect.width();
            else if( relPos.alignment() & Qt::AlignHCenter )
                pt.rx() -= 0.5 * boundRect.width();

            if( relPos.alignment() & Qt::AlignTop )
                pt.ry() += boundRect.height();
            else if( relPos.alignment() & Qt::AlignVCenter )
                pt.ry() += 0.5 * boundRect.height();
        }

        // FIXME draw the non-text bits, background, etc

        if ( a.showRepetitiveDataLabels() ||
             pos.x() <= d->lastX ||
             d->lastRoundedValue != roundedValue ) {
            d->lastRoundedValue = roundedValue;
            d->lastX = pos.x();

            PainterSaver painterSaver( painter );
            painter->setPen( ta.pen() );
            painter->setFont( calculatedFont );
            painter->translate( pt );
            painter->rotate( ta.rotation() );
            painter->drawText( QPointF(0, 0), roundedValue );
        }
    }
}
```

7.60.2.41 void AbstractDiagram::paintDataValueTexts (QPainter * *painter*) [protected, virtual, inherited]

Definition at line 576 of file KDChartAbstractDiagram.cpp.

```
                                                                               {
       for ( int j=0; j< rowCount; ++j ) {
           const QModelIndex index = model()->index( j, i, rootIndex() );
           double value = model()->data( index ).toDouble();
           const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
           paintDataValueText( painter, index, pos, value );
       }
    }
}
```

7.60.2.42 void RingDiagram::paintEvent (QPaintEvent ∗) `[protected]`

Definition at line 88 of file KDChartRingDiagram.cpp.

References paint(), KDChart::PaintContext::setPainter(), and KDChart::PaintContext::setRectangle().

```
{
    QPainter painter ( viewport() );
    PaintContext ctx;
    ctx.setPainter ( &painter );
    ctx.setRectangle( QRectF ( 0, 0, width(), height() ) );
    paint ( &ctx );
}
```

7.60.2.43 void AbstractDiagram::paintMarker (QPainter ∗ *painter*, const QModelIndex & *index*, const QPointF & *pos*) `[inherited]`

Definition at line 592 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::brush(), KDChart::AbstractDiagram::checkInvariants(), KDChart::AbstractDiagram::dataValueAttributes(), KDChart::MarkerAttributes::isVisible(), KDChart::DataValueAttributes::isVisible(), KDChart::DataValueAttributes::markerAttributes(), KDChart::MarkerAttributes::markerColor(), KDChart::MarkerAttributes::markerSize(), KDChart::AbstractDiagram::paintMarker(), and KDChart::MarkerAttributes::pen().

```
{

    if ( !checkInvariants() ) return;
    DataValueAttributes a = dataValueAttributes(index);
    if ( !a.isVisible() ) return;
    const MarkerAttributes &ma = a.markerAttributes();
    if ( !ma.isVisible() ) return;

    PainterSaver painterSaver( painter );
    QSizeF maSize( ma.markerSize() );
    QBrush indexBrush( brush( index ) );
    QPen indexPen( ma.pen() );
    if ( ma.markerColor().isValid() )
        indexBrush.setColor( ma.markerColor() );

    paintMarker( painter, ma, indexBrush, indexPen, pos, maSize );
}


```

7.60.2.44 void AbstractDiagram::paintMarker (QPainter ∗ *painter*, const MarkerAttributes & *markerAttributes*, const QBrush & *brush*, const QPen &, const QPointF & *point*, const QSizeF & *size*) `[virtual, inherited]`

Definition at line 614 of file KDChartAbstractDiagram.cpp.

References KDChart::MarkerAttributes::markerStyle().

Referenced by KDChart::MarkerLayoutItem::paintIntoRect(), and KDChart::AbstractDiagram::paintMarker().

```
{

    const QPen oldPen( painter->pen() );
    // Pen is used to paint 4Pixels - 1 Pixel - Ring and FastCross types.
    // make sure to use the brush color - see above in those cases.
    const bool isFourPixels = (markerAttributes.markerStyle() == MarkerAttributes::Marker4Pixels);
    if( isFourPixels || (markerAttributes.markerStyle() == MarkerAttributes::Marker1Pixel) ){
        // for high-performance point charts with tiny point markers:
        painter->setPen( QPen( brush.color().light() ) );
        if( isFourPixels ){
            const qreal x = pos.x();
            const qreal y = pos.y();
            painter->drawLine( QPointF(x-1.0,y-1.0),
                               QPointF(x+1.0,y-1.0) );
            painter->drawLine( QPointF(x-1.0,y),
                               QPointF(x+1.0,y) );
            painter->drawLine( QPointF(x-1.0,y+1.0),
                               QPointF(x+1.0,y+1.0) );
        }
        painter->drawPoint( pos );
    }else{
        PainterSaver painterSaver( painter );
        // we only a solid line surrounding the markers
        QPen painterPen( pen );
        painterPen.setStyle( Qt::SolidLine );
        painter->setPen( painterPen );
        painter->setBrush( brush );
        painter->setRenderHint ( QPainter::Antialiasing );
        painter->translate( pos );
        switch ( markerAttributes.markerStyle() ) {
            case MarkerAttributes::MarkerCircle:
                painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                            maSize.height(), maSize.width()) );
                break;
            case MarkerAttributes::MarkerSquare:
                {
                    QRectF rect( 0 - maSize.width()/2, 0 - maSize.height()/2,
                                maSize.width(), maSize.height() );
                    painter->drawRect( rect );
                    painter->fillRect( rect, brush.color() );
                    break;
                }
            case MarkerAttributes::MarkerDiamond:
                {
                    QVector <QPointF > diamondPoints;
                    QPointF top, left, bottom, right;
                    top    = QPointF( 0, 0 - maSize.height()/2 );
                    left   = QPointF( 0 - maSize.width()/2, 0 );
                    bottom = QPointF( 0, maSize.height()/2 );
                    right  = QPointF( maSize.width()/2, 0 );
                    diamondPoints << top << left << bottom << right;
                    painter->drawPolygon( diamondPoints );
                    break;
                }
            // both handled on top of the method:
            case MarkerAttributes::Marker1Pixel:
            case MarkerAttributes::Marker4Pixels:
                    break;
            case MarkerAttributes::MarkerRing:
                {
                    painter->setPen( QPen( brush.color() ) );
                    painter->setBrush( Qt::NoBrush );
                    painter->drawEllipse( QRectF( 0 - maSize.height()/2, 0 - maSize.width()/2,
                                        maSize.height(), maSize.width()) );
                    break;
                }
            case MarkerAttributes::MarkerCross:
                {
                    QRectF rect( maSize.width()*-0.5, maSize.height()*-0.2,
                                 maSize.width(), maSize.height()*0.4 );
                    painter->drawRect( rect );
                    rect.setTopLeft(QPointF( maSize.width()*-0.2, maSize.height()*-0.5 ));
                    rect.setSize(QSizeF( maSize.width()*0.4, maSize.height() ));
                    painter->drawRect( rect );
                    break;
                }
            case MarkerAttributes::MarkerFastCross:
                {
                    QPointF left, right, top, bottom;
                    left  = QPointF( -maSize.width()/2, 0 );
                    right = QPointF( maSize.width()/2, 0 );
                    top   = QPointF( 0, -maSize.height()/2 );
                    bottom= QPointF( 0, maSize.height()/2 );
                    painter->setPen( QPen( brush.color() ) );
                    painter->drawLine( left, right );
                    painter->drawLine(  top, bottom );
                    break;
                }
            default:
                Q_ASSERT_X ( false, "paintMarkers()",
                            "Type item does not match a defined Marker Type." );
        }
    }
    painter->setPen( oldPen );
}

void AbstractDiagram::paintMarkers( QPainter* painter )
```

7.60.2.45 void AbstractDiagram::paintMarkers (QPainter ∗ *painter*) [protected, virtual, inherited]

Definition at line 716 of file KDChartAbstractDiagram.cpp.

```
                                                                                {
       for ( int j=0; j< rowCount; ++j ) {
           const QModelIndex index = model()->index( j, i, rootIndex() );
           double value = model()->data( index ).toDouble();
           const QPointF pos = coordinatePlane()->translate( QPointF( j, value ) );
           paintMarker( painter, index, pos );
       }
    }
}


```

7.60.2.46 QPen AbstractDiagram::pen (const QModelIndex & *index*) const [inherited]

Retrieve the pen to be used, for painting the datapoint at the given index in the model.

Parameters:

index The index of the datapoint in the model.

Returns:

The pen to use for painting.

Definition at line 770 of file KDChartAbstractDiagram.cpp.

```
{
```

7.60.2.47 QPen AbstractDiagram::pen (int *dataset*) const `[inherited]`

Retrieve the pen to be used for the given dataset.

This will fall back automatically to what was set at model level, if there are no dataset specific settings.

Parameters:

dataset The dataset to retrieve the pen for.

Returns:

The pen to use for painting.

Definition at line 762 of file KDChartAbstractDiagram.cpp.

```
{
```

7.60.2.48 QPen AbstractDiagram::pen () const `[inherited]`

Retrieve the pen to be used for painting datapoints globally.

This will fall back automatically to the default settings, if there are no specific settings.

Returns:

The pen to use for painting.

Definition at line 756 of file KDChartAbstractDiagram.cpp.

Referenced by KDChart::PieDiagram::paint(), and KDChart::LineDiagram::paint().

```
{
```

7.60.2.49 bool AbstractDiagram::percentMode () const `[inherited]`

Definition at line 468 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::CartesianCoordinatePlane::getDataDimensionsList().

7.60.2.50 PieAttributes AbstractPieDiagram::pieAttributes (const QModelIndex & *index*) const `[inherited]`

Definition at line 121 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::PieAttributesRole.

```
{
    return qVariantValue<PieAttributes>(
        d->attributesModel->data(
            d->attributesModel->mapFromSource( index ),
            PieAttributesRole ) );
}
```

7.60.2.51 PieAttributes AbstractPieDiagram::pieAttributes (int *column*) const `[inherited]`

Definition at line 113 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::PieAttributesRole.

```
{
    return qVariantValue<PieAttributes>(
        d->attributesModel->data(
            d->attributesModel->mapFromSource( columnToIndex( column ) ).column(),
            PieAttributesRole ) );
}
```

7.60.2.52 PieAttributes AbstractPieDiagram::pieAttributes () const `[inherited]`

Definition at line 104 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::PieAttributesRole.

Referenced by KDChart::PieDiagram::calculateDataBoundaries(), and KDChart::PieDiagram::paint().

```
{
    return qVariantValue<PieAttributes>(
        d->attributesModel->data( PieAttributesRole ) );
}
```

7.60.2.53 const PolarCoordinatePlane ∗ AbstractPolarDiagram::polarCoordinatePlane () const `[inherited]`

Definition at line 55 of file KDChartAbstractPolarDiagram.cpp.

References KDChart::AbstractDiagram::coordinatePlane().

Referenced by KDChart::PieDiagram::paint().

```
{
    return dynamic_cast<const PolarCoordinatePlane*>( coordinatePlane() );
}
```

7.60.2.54 void KDChart::AbstractDiagram::propertiesChanged () `[protected, inherited]`

Emitted upon change of a property of the Diagram.

Referenced by KDChart::LineDiagram::resetLineAttributes(), KDChart::AbstractDiagram::setDataValueAttributes(), KDChart::LineDiagram::setLineAttributes(), KDChart::LineDiagram::setThreeDLineAttributes(), and KDChart::LineDiagram::setType().

7.60.2.55 bool RingDiagram::relativeThickness () const

Definition at line 74 of file KDChartRingDiagram.cpp.

References d.

```
{
    return d->relativeThickness;
}
```

7.60.2.56 void RingDiagram::resize (const QSizeF & *area*) `[virtual]`

[reimplemented]

Implements KDChart::AbstractDiagram.

Definition at line 128 of file KDChartRingDiagram.cpp.

```
129 {
130 }
```

7.60.2.57 void RingDiagram::resizeEvent (QResizeEvent ∗) `[protected]`

Definition at line 97 of file KDChartRingDiagram.cpp.

```
98 {
99 }
```

7.60.2.58 void AbstractDiagram::scrollTo (const QModelIndex & *index*, ScrollHint *hint* = EnsureVisible) `[virtual, inherited]`

[reimplemented]

Definition at line 830 of file KDChartAbstractDiagram.cpp.

```
832 { return QModelIndex(); }
```

7.60.2.59 void AbstractDiagram::setAllowOverlappingDataValueTexts (bool *allow*) `[inherited]`

Set whether data value labels are allowed to overlap.

Parameters:

allow True means that overlapping labels are allowed.

Definition at line 440 of file KDChartAbstractDiagram.cpp.

References d.

```
445 {
```

7.60.2.60 void AbstractDiagram::setAntiAliasing (bool *enabled*) `[inherited]`

Set whether anti-aliasing is to be used while rendering this diagram.

Parameters:

enabled True means that AA is enabled.

Definition at line 451 of file KDChartAbstractDiagram.cpp.

References d.

```
456 {
```

7.60.2.61 void AbstractDiagram::setAttributesModel (AttributesModel ∗ *model*) `[virtual, inherited]`

Associate an AttributesModel with this diagram.

Note that the diagram does _not_ take ownership of the AttributesModel. This should thus only be used with AttributesModels that have been explicitely created by the user, and are owned by her. Setting an AttributesModel that is internal to another diagram is an error.

Correct:

```
AttributesModel *am = new AttributesModel( model, 0 );
diagram1->setAttributesModel( am );
diagram2->setAttributesModel( am );
```

Wrong:

```
diagram1->setAttributesModel( diagram2->attributesModel() );
```

Parameters:

model The AttributesModel to use for this diagram.

See also:

AttributesModel, usesExternalAttributesModel

Definition at line 261 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::AbstractDiagram::modelsChanged().

```
{
    if( amodel->sourceModel() != model() ) {
        qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
                 "Trying to set an attributesmodel which works on a different "
                 "model than the diagram.");
        return;
    }
    if( qobject_cast<PrivateAttributesModel*>(amodel) ) {
        qWarning("KDChart::AbstractDiagram::setAttributesModel() failed: "
                 "Trying to set an attributesmodel that is private to another diagram.");
        return;
    }
    d->setAttributesModel(amodel);
    scheduleDelayedItemsLayout();
    d->databoundariesDirty = true;
    emit modelsChanged();
}
```

7.60.2.62 void AbstractDiagram::setAttributesModelRootIndex (const QModelIndex & *idx*) `[protected, inherited]`

Definition at line 301 of file KDChartAbstractDiagram.cpp.

References d.

7.60.2.63 void AbstractDiagram::setBrush (const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting all datasets in the model.

Parameters:

brush The brush to use.

Definition at line 786 of file KDChartAbstractDiagram.cpp.

```
792 {
```

7.60.2.64 void AbstractDiagram::setBrush (int *dataset*, const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting the given dataset.

Parameters:

dataset The dataset's column in the model.

pen The brush to use.

Definition at line 793 of file KDChartAbstractDiagram.cpp.

```
801 {
```

7.60.2.65 void AbstractDiagram::setBrush (const QModelIndex & *index*, const QBrush & *brush*) `[inherited]`

Set the brush to be used, for painting the datapoint at the given index.

Parameters:

index The datapoint's index in the model.

brush The brush to use.

Definition at line 778 of file KDChartAbstractDiagram.cpp.

```
785 {
```

7.60.2.66 void AbstractDiagram::setCoordinatePlane (AbstractCoordinatePlane ∗ *plane*) `[virtual, inherited]`

Set the coordinate plane associated with the diagram.

This determines how coordinates in value space are mapped into pixel space. The chart takes ownership.

Returns:

The coordinate plane associated with the diagram.

Reimplemented in KDChart::AbstractCartesianDiagram.

Definition at line 324 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractCoordinatePlane::addDiagram(), KDChart::AbstractCartesianDiagram::setCoordinatePlane(), and KDChart::AbstractCoordinatePlane::takeDiagram().

```
328 {
```

7.60.2.67 void AbstractDiagram::setDataBoundariesDirty () const `[protected, inherited]`

Definition at line 240 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::BarDiagram::setThreeDBarAttributes(), KDChart::LineDiagram::setThreeDLineAttributes(), KDChart::LineDiagram::setType(), and KDChart::BarDiagram::setType().

```
241 {
242     d->databoundariesDirty = true;
243 }
```

7.60.2.68 void AbstractDiagram::setDatasetDimension (int *dimension*) `[inherited]`

Sets the dataset dimension of the diagram.

See also:
datasetDimension.

Parameters:
dimension

Definition at line 947 of file KDChartAbstractDiagram.cpp.

References d.

```
954 {
```

7.60.2.69 void AbstractDiagram::setDataValueAttributes (const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for all datapoints in the model.

Parameters:
a The attributes to set.

Definition at line 434 of file KDChartAbstractDiagram.cpp.

References d.

```
439 {
```

7.60.2.70 void AbstractDiagram::setDataValueAttributes (int *dataset*, const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for the given dataset.

Parameters:
dataset The dataset to set the attributes for.

a The attributes to set.

Definition at line 406 of file KDChartAbstractDiagram.cpp.

References d.

```
413 {
```

7.60.2.71 void AbstractDiagram::setDataValueAttributes (const QModelIndex & *index*, const DataValueAttributes & *a*) `[inherited]`

Set the DataValueAttributes for the given index.

Parameters:

index The datapoint to set the attributes for.

a The attributes to set.

Definition at line 395 of file KDChartAbstractDiagram.cpp.

References d, KDChart::DataValueLabelAttributesRole, and KDChart::AbstractDiagram::propertiesChanged().

```
395 {
396     d->attributesModel->setData(
397         d->attributesModel->mapFromSource( index ),
398         qVariantFromValue( a ),
399         DataValueLabelAttributesRole );
400     emit propertiesChanged();
401 }
402
403
```

7.60.2.72 void AbstractPieDiagram::setGranularity (qreal *value*) `[inherited]`

Set the granularity: the smaller the granularity the more your diagram segments will show facettes instead of rounded segments.

Parameters:

value the granularity value between 0.05 (one twentieth of a degree) and 36.0 (one tenth of a full circle), other values will be interpreted as 1.0.

Definition at line 64 of file KDChartAbstractPieDiagram.cpp.

References d.

```
65 {
66     d->granularity = value;
67 }
```

7.60.2.73 void AbstractDiagram::setHidden (bool *hidden*) `[inherited]`

Hide (or unhide, resp.) all datapoints in the model.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

hidden The hidden status to set.

Definition at line 365 of file KDChartAbstractDiagram.cpp.

References d.

```
372 {
```

7.60.2.74 void AbstractDiagram::setHidden (int *column*, bool *hidden*) `[inherited]`

Hide (or unhide, resp.) a dataset.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

dataset The dataset to set the hidden status for.

hidden The hidden status to set.

Definition at line 356 of file KDChartAbstractDiagram.cpp.

References d.

```
364 {
```

7.60.2.75 void AbstractDiagram::setHidden (const QModelIndex & *index*, bool *hidden*) `[inherited]`

Hide (or unhide, resp.) a data cell.

Note:

Hidden data are still taken into account by the coordinate plane, so neither the grid nor your axes' ranges will change, when you hide data. For totally removing data from KD Chart's view you can use another approach: e.g. you could define a proxy model on top of your data model, and register the proxy model calling setModel() instead of registering your real data model.

Parameters:

index The datapoint to set the hidden status for.

hidden The hidden status to set.

Definition at line 347 of file KDChartAbstractDiagram.cpp.

References d, and KDChart::DataHiddenRole.

```
355 {
```

7.60.2.76 void AbstractDiagram::setModel (QAbstractItemModel ∗ *model*) `[virtual, inherited]`

Associate a model with the diagram.

Definition at line 245 of file KDChartAbstractDiagram.cpp.

References d, KDChart::AttributesModel::initFrom(), and KDChart::AbstractDiagram::modelsChanged().

```
246 {
247   QAbstractItemView::setModel( newModel );
248   AttributesModel* amodel = new PrivateAttributesModel( newModel, this );
249   amodel->initFrom( d->attributesModel );
250   d->setAttributesModel(amodel);
251   scheduleDelayedItemsLayout();
252   d->databoundariesDirty = true;
253   emit modelsChanged();
254 }
```

7.60.2.77 void AbstractDiagram::setPen (const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting all datasets in the model.

Parameters:

pen The pen to use.

Definition at line 740 of file KDChartAbstractDiagram.cpp.

```
746 {
```

7.60.2.78 void AbstractDiagram::setPen (int *dataset*, const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting the given dataset.

Parameters:

dataset The dataset's row in the model.

pen The pen to use.

Definition at line 747 of file KDChartAbstractDiagram.cpp.

```
755 {
```

7.60.2.79 void AbstractDiagram::setPen (const QModelIndex & *index*, const QPen & *pen*) `[inherited]`

Set the pen to be used, for painting the datapoint at the given index.

Parameters:

- ***index*** The datapoint's index in the model.
- ***pen*** The pen to use.

Definition at line 732 of file KDChartAbstractDiagram.cpp.

```
739 {
```

7.60.2.80 void AbstractDiagram::setPercentMode (bool *percent*) `[inherited]`

Definition at line 462 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::LineDiagram::setType(), and KDChart::BarDiagram::setType().

```
467 {
```

7.60.2.81 void AbstractPieDiagram::setPieAttributes (int *column*, const PieAttributes & *a*) `[inherited]`

Definition at line 94 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::PieAttributesRole.

```
95 {
96     d->attributesModel->setHeaderData(
97         column, Qt::Vertical, qVariantFromValue( attrs ), PieAttributesRole );
98     emit layoutChanged( this );
99 }
```

7.60.2.82 void AbstractPieDiagram::setPieAttributes (const PieAttributes & *a*) `[inherited]`

Definition at line 88 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::PieAttributesRole.

```
89 {
90     d->attributesModel->setModelData( qVariantFromValue( attrs ), PieAttributesRole );
91     emit layoutChanged( this );
92 }
```

7.60.2.83 void RingDiagram::setRelativeThickness (bool *relativeThickness*)

Definition at line 69 of file KDChartRingDiagram.cpp.

References d.

```
{
    d->relativeThickness = relativeThickness;
}
```

7.60.2.84 void AbstractDiagram::setRootIndex (const QModelIndex & *idx*) `[virtual, inherited]`

Set the root index in the model, where the diagram starts referencing data for display.

[reimplemented]

Definition at line 294 of file KDChartAbstractDiagram.cpp.

References d.

7.60.2.85 void AbstractDiagram::setSelection (const QRect & *rect*, QItemSelectionModel::SelectionFlags *command*) `[virtual, inherited]`

[reimplemented]

Definition at line 848 of file KDChartAbstractDiagram.cpp.

```
{ return QRegion(); }
```

7.60.2.86 void AbstractPieDiagram::setStartPosition (int *degrees*) `[inherited]`

Deprecated

Use PolarCoordinatePlane::setStartPosition(qreal degrees) instead.

Definition at line 77 of file KDChartAbstractPieDiagram.cpp.

```
{
    qWarning() << "Deprecated AbstractPieDiagram::setStartPosition() called, setting ignored.";
}
```

7.60.2.87 void AbstractPieDiagram::setThreeDPieAttributes (const QModelIndex & *index*, const ThreeDPieAttributes & *a*) `[inherited]`

Definition at line 143 of file KDChartAbstractPieDiagram.cpp.

References KDChart::ThreeDPieAttributesRole.

```
{
    model()->setData( index, qVariantFromValue( tda ), ThreeDPieAttributesRole );
    emit layoutChanged( this );
}
```

7.60.2.88 void AbstractPieDiagram::setThreeDPieAttributes (int *column*, const ThreeDPieAttributes & *a*) `[inherited]`

Definition at line 136 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::ThreeDPieAttributesRole.

```
{
    d->attributesModel->setHeaderData(
        column, Qt::Vertical, qVariantFromValue( tda ), ThreeDPieAttributesRole );
    emit layoutChanged( this );
}
```

7.60.2.89 void AbstractPieDiagram::setThreeDPieAttributes (const ThreeDPieAttributes & *a*) `[inherited]`

Definition at line 130 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::ThreeDPieAttributesRole.

```
{
    d->attributesModel->setModelData( qVariantFromValue( tda ), ThreeDPieAttributesRole );
    emit layoutChanged( this );
}
```

7.60.2.90 int AbstractPieDiagram::startPosition () const `[inherited]`

Deprecated

Use qreal PolarCoordinatePlane::startPosition instead.

Definition at line 82 of file KDChartAbstractPieDiagram.cpp.

```
{
    qWarning() << "Deprecated AbstractPieDiagram::startPosition() called.";
    return 0;
}
```

7.60.2.91 ThreeDPieAttributes AbstractPieDiagram::threeDPieAttributes (const QModelIndex & *index*) const `[inherited]`

Definition at line 169 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::ThreeDPieAttributesRole.

```
{
    return qVariantValue<ThreeDPieAttributes>(
        d->attributesModel->data(
            d->attributesModel->mapFromSource( index ),
            ThreeDPieAttributesRole ) );
}
```

7.60.2.92 ThreeDPieAttributes AbstractPieDiagram::threeDPieAttributes (int *column*) const `[inherited]`

Definition at line 161 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::ThreeDPieAttributesRole.

```
{
    return qVariantValue<ThreeDPieAttributes>(
        d->attributesModel->data(
            d->attributesModel->mapFromSource( columnToIndex( column ) ).column(),
            ThreeDPieAttributesRole ) );
}
```

7.60.2.93 ThreeDPieAttributes AbstractPieDiagram::threeDPieAttributes () const `[inherited]`

Definition at line 152 of file KDChartAbstractPieDiagram.cpp.

References d, and KDChart::ThreeDPieAttributesRole.

Referenced by KDChart::PieDiagram::paint().

```
{
    return qVariantValue<ThreeDPieAttributes>(
        d->attributesModel->data( ThreeDPieAttributesRole ) );
}
```

7.60.2.94 void AbstractDiagram::update () const `[inherited]`

Definition at line 961 of file KDChartAbstractDiagram.cpp.

References d.

Referenced by KDChart::AbstractDiagram::doItemsLayout().

7.60.2.95 void KDChart::AbstractDiagram::useDefaultColors () `[inherited]`

Set the palette to be used, for painting datasets to the default palette.

See also:

KDChart::Palette. FIXME: fold into one usePalette (KDChart::Palette&) method

Definition at line 855 of file KDChartAbstractDiagram.cpp.

References d.

```
{
```

7.60.2.96 void KDChart::AbstractDiagram::useRainbowColors () `[inherited]`

Set the palette to be used, for painting datasets to the rainbow palette.

See also:

KDChart::Palette.

Definition at line 865 of file KDChartAbstractDiagram.cpp.

References d.

```
869 {
```

7.60.2.97 bool AbstractDiagram::usesExternalAttributesModel () const `[virtual, inherited]`

Returns whether the diagram is using its own built-in attributes model or an attributes model that was set via setAttributesModel.

See also:

setAttributesModel

Definition at line 280 of file KDChartAbstractDiagram.cpp.

References d.

```
281 {
282     return d->usesExternalAttributesModel();
283 }
```

7.60.2.98 void KDChart::AbstractDiagram::useSubduedColors () `[inherited]`

Set the palette to be used, for painting datasets to the subdued palette.

See also:

KDChart::Palette.

Definition at line 860 of file KDChartAbstractDiagram.cpp.

References d.

```
864 {
```

7.60.2.99 double AbstractDiagram::valueForCell (int *row*, int *column*) const `[protected, inherited]`

Helper method, retrieving the data value (DisplayRole) for a given row and column.

Parameters:

row The row to query.

column The column to query.

Returns:

The value of the display role at the given row and column as a double.

Definition at line 955 of file KDChartAbstractDiagram.cpp.

References KDChart::AbstractDiagram::attributesModelRootIndex(), and d.

Referenced by KDChart::LineDiagram::paint().

```
960 {
```

7.60.2.100 double RingDiagram::valueTotals () const `[virtual]`

[reimplemented]

Implements KDChart::AbstractPolarDiagram.

Definition at line 133 of file KDChartRingDiagram.cpp.

```
134 {
135     double total = 0;
136     const int colCount = model()->columnCount(rootIndex());
137     for ( int j=0; j<colCount; ++j ) {
138       total += model()->data( model()->index( 0, j, rootIndex() ) ).toDouble();
139     }
140     return total;
141 }
```

7.60.2.101 int AbstractDiagram::verticalOffset () const `[virtual, inherited]`

[reimplemented]

Definition at line 842 of file KDChartAbstractDiagram.cpp.

```
844 { return true; }
```

7.60.2.102 QRect AbstractDiagram::visualRect (const QModelIndex & *index*) const `[virtual, inherited]`

[reimplemented]

Definition at line 825 of file KDChartAbstractDiagram.cpp.

```
829 {}
```

7.60.2.103 QRegion AbstractDiagram::visualRegionForSelection (const QItemSelection & *selection*) const `[virtual, inherited]`

[reimplemented]

Definition at line 851 of file KDChartAbstractDiagram.cpp.

7.60.3 Member Data Documentation

7.60.3.1 Q_SIGNALS KDChart::AbstractDiagram::__pad0__ `[protected, inherited]`

Definition at line 589 of file KDChartAbstractDiagram.h.

The documentation for this class was generated from the following files:

- KDChartRingDiagram.h
- KDChartRingDiagram.cpp

7.61 KDChart::SignalCompressor Class Reference

`#include <KDChartSignalCompressor.h>`

Inheritance diagram for KDChart::SignalCompressor:Collaboration diagram for KDChart::SignalCompressor:

7.61.1 Detailed Description

SignalCompressor compresses signals where the same signal needs to be emitted by several pieces of the code, but only one of the signals should be received at the end.

Usage: create a object of SignalCompressor, and give it the name and object of the signal it is supposed to manage instead of emitting the signal, call emitSignal() on the compressor the signal will only be emitted once, and that is after the current call stack ends and returns to the event loop

With the current implementation, the class changes the sematics of signals to be a queued connection. If that is not wanted, another compression algorithm needs to be implemented. Also, at the moment, only nullary signals are supported, as parameters could not be compressed. A typical use of the class is to compress update notifications. This class is not part of the published KDChart API.

Definition at line 29 of file KDChartSignalCompressor.h.

Public Member Functions

- SignalCompressor (QObject ∗receiver, const char ∗signal, QObject ∗parent=0)

Public Attributes

- Q_SIGNALS __pad0__: void finallyEmit()
- private Q_SLOTS: void nowGoAlready()
- public Q_SLOTS: void emitSignal()

7.61.2 Constructor & Destructor Documentation

7.61.2.1 SignalCompressor::SignalCompressor (QObject ∗ *receiver*, const char ∗ *signal*, QObject ∗ *parent* = 0)

Definition at line 5 of file KDChartSignalCompressor.cpp.

```
    : QObject( parent )
{
    connect( this, SIGNAL( finallyEmit() ), receiver, signal );
     connect( &m_timer, SIGNAL( timeout() ), SLOT( nowGoAlready() ) );
     m_timer.setSingleShot( true );
     // m_timer.setIntervall( 0 ); // default, just to know...
}
```

7.61.3 Member Data Documentation

7.61.3.1 Q_SIGNALS KDChart::SignalCompressor::__pad0__

Definition at line 38 of file KDChartSignalCompressor.h.

7.61.3.2 private KDChart::SignalCompressor::Q_SLOTS

Definition at line 44 of file KDChartSignalCompressor.h.

7.61.3.3 public KDChart::SignalCompressor::Q_SLOTS

Definition at line 41 of file KDChartSignalCompressor.h.

The documentation for this class was generated from the following files:

- KDChartSignalCompressor.h
- KDChartSignalCompressor.cpp

7.62 KDChart::TextArea Class Reference

`#include <KDChartTextArea.h>`

Inheritance diagram for KDChart::TextArea:Collaboration diagram for KDChart::TextArea:

7.62.1 Detailed Description

A text area in the chart with a background, a frame, etc.

TextArea is the base class for all text containing non-widget chart elements that have a set of background attributes and frame attributes, such as headers or footers.

Note:

> This class inherits from AbstractAreaBase, TextLayoutItem, QObject. The reason for this tripple inheritance is that neither AbstractAreaBase nor TextLayoutItem are QObject.

Definition at line 54 of file KDChartTextArea.h.

Public Member Functions

- void alignToReferencePoint (const RelativePosition &position)
- const QObject ∗ autoReferenceArea () const
- BackgroundAttributes backgroundAttributes () const
- bool compare (const AbstractAreaBase ∗other) const

  Returns true if both areas have the same settings.

- virtual Qt::Orientations expandingDirections () const

  pure virtual in QLayoutItem

- FrameAttributes frameAttributes () const
- virtual QRect geometry () const

  pure virtual in QLayoutItem

- void getFrameLeadings (int &left, int &top, int &right, int &bottom) const
- virtual bool intersects (const TextLayoutItem &other, const QPoint &myPos, const QPoint &otherPos) const
- virtual bool intersects (const TextLayoutItem &other, const QPointF &myPos, const QPointF &otherPos) const
- virtual bool isEmpty () const

  pure virtual in QLayoutItem

- virtual QSize maximumSize () const

  pure virtual in QLayoutItem

- virtual QSize minimumSize () const

  pure virtual in QLayoutItem

- virtual void paint (QPainter ∗)
- void paintAll (QPainter &painter)

Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

- virtual void paintBackground (QPainter &painter, const QRect &rectangle)
- virtual void paintCtx (PaintContext ∗context)

  Default impl: Paint the complete item using its layouted position and size.

- virtual void paintFrame (QPainter &painter, const QRect &rectangle)
- virtual void paintIntoRect (QPainter &painter, const QRect &rect)

  Draws the background and frame, then calls paint().

- QLayout ∗ parentLayout ()
- virtual QFont realFont () const
- virtual qreal realFontSize () const
- void removeFromParentLayout ()
- void setAutoReferenceArea (const QObject ∗area)
- void setBackgroundAttributes (const BackgroundAttributes &a)
- void setFrameAttributes (const FrameAttributes &a)
- virtual void setGeometry (const QRect &r)

  pure virtual in QLayoutItem

- void setParentLayout (QLayout ∗lay)
- virtual void setParentWidget (QWidget ∗widget)

  Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

- void setText (const QString &text)
- void setTextAttributes (const TextAttributes &a)

  Use this to specify the text attributes to be used for this item.

- virtual QSize sizeHint () const

  pure virtual in QLayoutItem

- virtual void sizeHintChanged () const

  Report changed size hint: ask the parent widget to recalculate the layout.

- QString text () const
- TextAttributes textAttributes () const

  Returns the text attributes to be used for this item.

- virtual ∼TextArea ()

Static Public Member Functions

- void paintBackgroundAttributes (QPainter &painter, const QRect &rectangle, const KDChart::BackgroundAttributes &attributes)
- void paintFrameAttributes (QPainter &painter, const QRect &rectangle, const KDChart::FrameAttributes &attributes)

Protected Member Functions

- virtual QRect areaGeometry () const
- QRect innerRect () const
- virtual void positionHasChanged ()
- TextArea ()

Protected Attributes

- Q_SIGNALS __pad0__: void positionChanged(TextArea ∗)
- QWidget ∗ mParent
- QLayout ∗ mParentLayout

7.62.2 Constructor & Destructor Documentation

7.62.2.1 TextArea::∼TextArea () `[virtual]`

Definition at line 60 of file KDChartTextArea.cpp.

```
{
    // this bloc left empty intentionally
}
```

7.62.2.2 TextArea::TextArea () `[protected]`

Definition at line 52 of file KDChartTextArea.cpp.

```
    : QObject()
    , KDChart::AbstractAreaBase()
    , KDChart::TextLayoutItem()
{
    // this bloc left empty intentionally
}
```

7.62.3 Member Function Documentation

7.62.3.1 void AbstractAreaBase::alignToReferencePoint (const RelativePosition & *position*) `[inherited]`

Definition at line 90 of file KDChartAbstractAreaBase.cpp.

```
{
    Q_UNUSED( position );
    // PENDING(kalle) FIXME
    qWarning( "Sorry, not implemented: void AbstractAreaBase::alignToReferencePoint( const RelativePosi
}
```

7.62.3.2 QRect TextArea::areaGeometry () const `[protected, virtual]`

Implements KDChart::AbstractAreaBase.

Definition at line 105 of file KDChartTextArea.cpp.

References KDChart::TextLayoutItem::geometry().

Referenced by paintAll().

```
{
    return geometry();
}
```

7.62.3.3 const QObject ∗ KDChart::TextLayoutItem::autoReferenceArea () const `[inherited]`

Definition at line 135 of file KDChartLayoutItems.cpp.

Referenced by KDChart::HeaderFooter::setParent().

```
{
    return mAutoReferenceArea;
}
```

7.62.3.4 BackgroundAttributes AbstractAreaBase::backgroundAttributes () const `[inherited]`

Definition at line 112 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by updateCommonBrush().

```
{
    return d->backgroundAttributes;
}
```

7.62.3.5 bool AbstractAreaBase::compare (const AbstractAreaBase ∗ *other*) const `[inherited]`

Returns true if both areas have the same settings.

Definition at line 75 of file KDChartAbstractAreaBase.cpp.

```
{
    if( other == this ) return true;
    if( ! other ){
        //qDebug() << "CartesianAxis::compare() cannot compare to Null pointer";
        return false;
    }
    /*
    qDebug() << "AbstractAreaBase:" << (frameAttributes() == other->frameAttributes())
        << (backgroundAttributes() == other->backgroundAttributes()) << "\n";
    */
    return  (frameAttributes()      == other->frameAttributes()) &&
            (backgroundAttributes() == other->backgroundAttributes());
}
```

7.62.3.6 Qt::Orientations KDChart::TextLayoutItem::expandingDirections () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 175 of file KDChartLayoutItems.cpp.

```
{
    return 0; // Grow neither vertically nor horizontally
}
```

7.62.3.7 FrameAttributes AbstractAreaBase::frameAttributes () const `[inherited]`

Definition at line 102 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone(), and updateCommonBrush().

```
{
    return d->frameAttributes;
}
```

7.62.3.8 QRect KDChart::TextLayoutItem::geometry () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 180 of file KDChartLayoutItems.cpp.

Referenced by areaGeometry(), KDChart::TextLayoutItem::paint(), paintAll(), KDChart::CartesianAxis::paintCtx(), and paintIntoRect().

```
{
    return mRect;
}
```

7.62.3.9 void AbstractAreaBase::getFrameLeadings (int & *left*, int & *top*, int & *right*, int & *bottom*) const `[inherited]`

Definition at line 204 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::AbstractAreaBase::innerRect(), and KDChart::AbstractAreaWidget::paintAll().

```
{
    if( d && d->frameAttributes.isVisible() ){
        const int padding = qMax( d->frameAttributes.padding(), 0 );
        left   = padding;
        top    = padding;
        right  = padding;
        bottom = padding;
    }else{
        left   = 0;
        top    = 0;
        right  = 0;
        bottom = 0;
    }
}
```

7.62.3.10 QRect AbstractAreaBase::innerRect () const `[protected, inherited]`

Definition at line 220 of file KDChartAbstractAreaBase.cpp.

References KDChart::AbstractAreaBase::areaGeometry(), and KDChart::AbstractAreaBase::getFrameLeadings().

Referenced by paintAll(), and KDChart::AbstractArea::paintAll().

```
{
    int left;
    int top;
    int right;
    int bottom;
    getFrameLeadings( left, top, right, bottom );
    return
        QRect( QPoint(0,0), areaGeometry().size() )
            .adjusted( left, top, -right, -bottom );
}
```

7.62.3.11 bool KDChart::TextLayoutItem::intersects (const TextLayoutItem & *other*, const QPoint & *myPos*, const QPoint & *otherPos*) const `[virtual, inherited]`

Definition at line 254 of file KDChartLayoutItems.cpp.

References KDChart::TextLayoutItem::mAttributes, PI, KDChart::TextLayoutItem::rotatedCorners(), KDChart::TextAttributes::rotation(), and KDChart::TextLayoutItem::unrotatedSizeHint().

```
{
    if ( mAttributes.rotation() != other.mAttributes.rotation() )
    {
        // that's the code for the common case: the rotation angles don't need to match here
        QPolygon myPolygon(          rotatedCorners() );
        QPolygon otherPolygon( other.rotatedCorners() );

        // move the polygons to their positions
        myPolygon.translate( myPos );
        otherPolygon.translate( otherPos );

        // create regions out of it
        QRegion myRegion( myPolygon );
        QRegion otherRegion( otherPolygon );

        // now the question - do they intersect or not?
        return ! myRegion.intersect( otherRegion ).isEmpty();

    } else {
        // and that's the code for the special case: the rotation angles match, which is less time cor
        const qreal angle = mAttributes.rotation() * PI / 180.0;
        // both sizes
        const QSizeF mySize(          unrotatedSizeHint() );
        const QSizeF otherSize( other.unrotatedSizeHint() );

        // that's myP1 relative to myPos
        QPointF myP1( mySize.height() * sin( angle ), 0.0 );
        // that's otherP1 to myPos
        QPointF otherP1 = QPointF( otherSize.height() * sin( angle ), 0.0 ) + otherPos - myPos;

        // now rotate both points the negative angle around myPos
        myP1 = QPointF( myP1.x() * cos( -angle ), myP1.x() * sin( -angle ) );
        qreal r = sqrt( otherP1.x() * otherP1.x() + otherP1.y() * otherP1.y() );
        otherP1 = QPointF( r * cos( -angle ), r * sin( -angle ) );

        // finally we look, whether both rectangles intersect or even not
        return QRectF( myP1, mySize ).intersects( QRectF( otherP1, otherSize ) );
    }
}
```

7.62.3.12 bool KDChart::TextLayoutItem::intersects (const TextLayoutItem & *other*, const QPointF & *myPos*, const QPointF & *otherPos*) const `[virtual, inherited]`

Definition at line 249 of file KDChartLayoutItems.cpp.

Referenced by KDChart::CartesianAxis::paintCtx().

```
{
    return intersects( other, myPos.toPoint(), otherPos.toPoint() );
}
```

7.62.3.13 bool KDChart::TextLayoutItem::isEmpty () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 185 of file KDChartLayoutItems.cpp.

```
{
    return false; // never empty, otherwise the layout item would not exist
}
```

7.62.3.14 QSize KDChart::TextLayoutItem::maximumSize () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 190 of file KDChartLayoutItems.cpp.

References KDChart::TextLayoutItem::sizeHint().

```
{
    return sizeHint(); // PENDING(kalle) Review, quite inflexible
}
```

7.62.3.15 QSize KDChart::TextLayoutItem::minimumSize () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 195 of file KDChartLayoutItems.cpp.

References KDChart::TextLayoutItem::sizeHint().

```
{
    return sizeHint(); // PENDING(kalle) Review, quite inflexible
}
```

7.62.3.16 void KDChart::TextLayoutItem::paint (QPainter ∗) `[virtual, inherited]`

Implements KDChart::AbstractLayoutItem.

Definition at line 382 of file KDChartLayoutItems.cpp.

References KDChart::TextLayoutItem::geometry(), KDChart::TextAttributes::pen(), rotatedRect(), and KDChart::TextAttributes::rotation().

Referenced by paintAll(), and KDChart::CartesianAxis::paintCtx().

```
383 {
384     // make sure, cached font is updated, if needed:
385     // sizeHint();
386 
387     if( !mRect.isValid() )
388         return;
389 
390     PainterSaver painterSaver( painter );
391     painter->setFont( cachedFont );
392     QRectF rect( geometry() );
393 
394 // #ifdef DEBUG_ITEMS_PAINT
395 //     painter->setPen( Qt::black );
396 //     painter->drawRect( rect );
397 // #endif
398     painter->translate( rect.center() );
399     rect.moveTopLeft( QPointF( - rect.width() / 2, - rect.height() / 2 ) );
400 #ifdef DEBUG_ITEMS_PAINT
401     painter->setPen( Qt::blue );
402     painter->drawRect( rect );
403 #endif
404     painter->rotate( mAttributes.rotation() );
405     rect = rotatedRect( rect, mAttributes.rotation() );
406 #ifdef DEBUG_ITEMS_PAINT
407     painter->setPen( Qt::red );
408     painter->drawRect( rect );
409 #endif
410     painter->setPen( mAttributes.pen() );
411     painter->drawText( rect, Qt::AlignHCenter | Qt::AlignVCenter, mText );
412 //    if (  calcSizeHint( cachedFont ).width() > rect.width() )
413 //        qDebug() << "rect.width()" << rect.width() << "text.width()" << calcSizeHint( cachedFont ).w
414 //
415 //    //painter->drawText( rect, Qt::AlignHCenter | Qt::AlignVCenter, mText );
416 }
```

7.62.3.17 void TextArea::paintAll (QPainter & *painter*) `[virtual]`

Call paintAll, if you want the background and the frame to be drawn before the normal paint() is invoked automatically.

Reimplemented from KDChart::AbstractLayoutItem.

Definition at line 83 of file KDChartTextArea.cpp.

References areaGeometry(), KDChart::TextLayoutItem::geometry(), KDChart::AbstractAreaBase::innerRect(), KDChart::TextLayoutItem::paint(), KDChart::AbstractAreaBase::paintBackground(), KDChart::AbstractAreaBase::paintFrame(), and KDChart::TextLayoutItem::setGeometry().

Referenced by paintIntoRect().

```
84 {
85     // Paint the background and frame
86     paintBackground( painter, geometry() );
87     paintFrame(      painter, geometry() );
88
89     // temporarily adjust the widget size, to be sure all content gets calculated
90     // to fit into the inner rectangle
91     const QRect oldGeometry( areaGeometry()  );
92     QRect inner( innerRect() );
93     inner.moveTo(
94         oldGeometry.left() + inner.left(),
95         oldGeometry.top()  + inner.top() );
96     const bool needAdjustGeometry = oldGeometry != inner;
97     if( needAdjustGeometry )
98         setGeometry( inner );
99     paint( &painter );
100     if( needAdjustGeometry )
101         setGeometry( oldGeometry );
102     //qDebug() << "TextAreaWidget::paintAll() done.";
103 }
```

7.62.3.18 void AbstractAreaBase::paintBackground (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 188 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintBackgroundAttributes().

Referenced by paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
189 {
190     Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintBackground()",
191                 "Private class was not initialized!" );
192     paintBackgroundAttributes( painter, rect, d->backgroundAttributes );
193 }
```

7.62.3.19 void AbstractAreaBase::paintBackgroundAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::BackgroundAttributes & *attributes*) `[static, inherited]`

Definition at line 119 of file KDChartAbstractAreaBase.cpp.

References KDChart::BackgroundAttributes::brush(), KDChart::BackgroundAttributes::isVisible(), KDChart::BackgroundAttributes::pixmap(), and KDChart::BackgroundAttributes::pixmapMode().

Referenced by KDChart::AbstractAreaBase::paintBackground().

```
121 {
122     if( !attributes.isVisible() ) return;
123
124     /* first draw the brush (may contain a pixmap)*/
125     if( Qt::NoBrush != attributes.brush().style() ) {
126         KDChart::PainterSaver painterSaver( &painter );
127         painter.setPen( Qt::NoPen );
128         const QPointF newTopLeft( painter.deviceMatrix().map( rect.topLeft() ) );
129         painter.setBrushOrigin( newTopLeft );
130         painter.setBrush( attributes.brush() );
131         painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
132     }
133     /* next draw the backPixmap over the brush */
134     if( !attributes.pixmap().isNull() &&
135         attributes.pixmapMode() != BackgroundAttributes::BackgroundPixmapModeNone ) {
136         QPointF ol = rect.topLeft();
137         if( BackgroundAttributes::BackgroundPixmapModeCentered == attributes.pixmapMode() )
138         {
139             ol.setX( rect.center().x() - attributes.pixmap().width() / 2 );
140             ol.setY( rect.center().y() - attributes.pixmap().height()/ 2 );
141             painter.drawPixmap( ol, attributes.pixmap() );
142         } else {
143             QMatrix m;
144             double zW = (double)rect.width()  / (double)attributes.pixmap().width();
145             double zH = (double)rect.height() / (double)attributes.pixmap().height();
146             switch( attributes.pixmapMode() ) {
147             case BackgroundAttributes::BackgroundPixmapModeScaled:
148             {
149                 double z;
150                 z = qMin( zW, zH );
151                 m.scale( z, z );
152             }
153             break;
154             case BackgroundAttributes::BackgroundPixmapModeStretched:
155                 m.scale( zW, zH );
156                 break;
157             default:
158                 ; // Cannot happen, previously checked
159             }
160             QPixmap pm = attributes.pixmap().transformed( m );
161             ol.setX( rect.center().x() - pm.width() / 2 );
162             ol.setY( rect.center().y() - pm.height()/ 2 );
163             painter.drawPixmap( ol, pm );
164         }
165     }
166 }
```

7.62.3.20 void KDChart::AbstractLayoutItem::paintCtx (PaintContext ∗ *context*) `[virtual, inherited]`

Default impl: Paint the complete item using its layouted position and size.

Reimplemented in KDChart::CartesianAxis.

Definition at line 77 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint(), and KDChart::PaintContext::painter().

```
78 {
79     if( context )
80         paint( context->painter() );
81 }
```

7.62.3.21 void AbstractAreaBase::paintFrame (QPainter & *painter*, const QRect & *rectangle*) `[virtual, inherited]`

Definition at line 196 of file KDChartAbstractAreaBase.cpp.

References d, and KDChart::AbstractAreaBase::paintFrameAttributes().

Referenced by paintAll(), KDChart::AbstractAreaWidget::paintAll(), and KDChart::AbstractArea::paintAll().

```
197 {
198     Q_ASSERT_X ( d != 0, "AbstractAreaBase::paintFrame()",
                "Private class was not initialized!" );
    paintFrameAttributes( painter, rect, d->frameAttributes );
}
```

7.62.3.22 void AbstractAreaBase::paintFrameAttributes (QPainter & *painter*, const QRect & *rectangle*, const KDChart::FrameAttributes & *attributes*) `[static, inherited]`

Definition at line 169 of file KDChartAbstractAreaBase.cpp.

References KDChart::FrameAttributes::isVisible(), and KDChart::FrameAttributes::pen().

Referenced by KDChart::AbstractAreaBase::paintFrame().

```
{

    if( !attributes.isVisible() ) return;

    // Note: We set the brush to NoBrush explicitely here.
    //       Otherwise we might get a filled rectangle, so any
    //       previously drawn background would be overwritten by that area.

    const QPen   oldPen(   painter.pen() );
    const QBrush oldBrush( painter.brush() );
    painter.setPen(   attributes.pen() );
    painter.setBrush( Qt::NoBrush );
    painter.drawRect( rect.adjusted( 0, 0, -1, -1 ) );
    painter.setBrush( oldBrush );
    painter.setPen(   oldPen );
}
```

7.62.3.23 void TextArea::paintIntoRect (QPainter & *painter*, const QRect & *rect*) `[virtual]`

Draws the background and frame, then calls paint().

In most cases there is no need to overwrite this method in a derived class, but you would overwrite TextLayoutItem::paint() instead.

Definition at line 71 of file KDChartTextArea.cpp.

References KDChart::TextLayoutItem::geometry(), paintAll(), and KDChart::TextLayoutItem::setGeometry().

```
{
    const QRect oldGeometry( geometry() );
    if( oldGeometry != rect )
        setGeometry( rect );
    painter.translate( rect.left(), rect.top() );
    paintAll( painter );
    painter.translate( -rect.left(), -rect.top() );
    if( oldGeometry != rect )
        setGeometry( oldGeometry );
}
```

7.62.3.24 QLayout∗ KDChart::AbstractLayoutItem::parentLayout () `[inherited]`

Definition at line 74 of file KDChartLayoutItems.h.

```
        {
            return mParentLayout;
        }
```

7.62.3.25 void TextArea::positionHasChanged () `[protected, virtual]`

Reimplemented from KDChart::AbstractAreaBase.

Definition at line 110 of file KDChartTextArea.cpp.

```
{
    emit positionChanged( this );
}
```

7.62.3.26 QFont KDChart::TextLayoutItem::realFont () const `[virtual, inherited]`

Definition at line 226 of file KDChartLayoutItems.cpp.

Referenced by KDChart::CartesianAxis::maximumSize(), and KDChart::CartesianAxis::paintCtx().

```
{
    realFontWasRecalculated(); // we can safely ignore the boolean return value
    return cachedFont;
}
```

7.62.3.27 qreal KDChart::TextLayoutItem::realFontSize () const `[virtual, inherited]`

Definition at line 206 of file KDChartLayoutItems.cpp.

References KDChart::TextAttributes::calculatedFontSize().

```
{
    return mAttributes.calculatedFontSize( mAutoReferenceArea, mAutoReferenceOrientation );
}
```

7.62.3.28 void KDChart::AbstractLayoutItem::removeFromParentLayout () `[inherited]`

Definition at line 78 of file KDChartLayoutItems.h.

Referenced by KDChart::Chart::takeCoordinatePlane().

```
        {
            if( mParentLayout ){
                if( widget() )
                    mParentLayout->removeWidget( widget() );
                else
                    mParentLayout->removeItem( this );
            }
        }
```

7.62.3.29 void KDChart::TextLayoutItem::setAutoReferenceArea (const QObject ∗ *area*) `[inherited]`

Definition at line 128 of file KDChartLayoutItems.cpp.

References KDChart::TextLayoutItem::sizeHint().

Referenced by KDChart::HeaderFooter::setParent().

```
129 {
130     mAutoReferenceArea = area;
131     cachedSizeHint = QSize();
132     sizeHint();
133 }
```

7.62.3.30 void AbstractAreaBase::setBackgroundAttributes (const BackgroundAttributes & *a*) `[inherited]`

Definition at line 107 of file KDChartAbstractAreaBase.cpp.

References d.

```
108 {
109     d->backgroundAttributes = a;
110 }
```

7.62.3.31 void AbstractAreaBase::setFrameAttributes (const FrameAttributes & *a*) `[inherited]`

Definition at line 97 of file KDChartAbstractAreaBase.cpp.

References d.

Referenced by KDChart::Legend::clone().

```
98 {
99     d->frameAttributes = a;
100 }
```

7.62.3.32 void KDChart::TextLayoutItem::setGeometry (const QRect & *r*) `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 200 of file KDChartLayoutItems.cpp.

Referenced by paintAll(), KDChart::CartesianAxis::paintCtx(), and paintIntoRect().

```
201 {
202     mRect = r;
203 }
```

7.62.3.33 void KDChart::AbstractLayoutItem::setParentLayout (QLayout ∗ *lay*) `[inherited]`

Definition at line 70 of file KDChartLayoutItems.h.

```
        {
            mParentLayout = lay;
        }
```

7.62.3.34 void KDChart::AbstractLayoutItem::setParentWidget (QWidget ∗ *widget*) `[virtual, inherited]`

Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

Thus, you need to call setParentWidget on every item, that has a non-fixed size.

Definition at line 64 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::mParent.

Referenced by KDChart::Legend::buildLegend(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
{
    mParent = widget;
}
```

7.62.3.35 void KDChart::TextLayoutItem::setText (const QString & *text*) `[inherited]`

Definition at line 140 of file KDChartLayoutItems.cpp.

References KDChart::TextLayoutItem::sizeHint().

Referenced by KDChart::Widget::addHeaderFooter(), KDChart::CartesianAxis::maximumSize(), and KDChart::CartesianAxis::paintCtx().

```
{
    mText = text;
    cachedSizeHint = QSize();
    sizeHint();
}
```

7.62.3.36 void KDChart::TextLayoutItem::setTextAttributes (const TextAttributes & *a*) `[inherited]`

Use this to specify the text attributes to be used for this item.

See also:

textAttributes

Definition at line 157 of file KDChartLayoutItems.cpp.

References KDChart::TextLayoutItem::sizeHint().

Referenced by KDChart::HeaderFooter::clone().

```
158 {
159     mAttributes = a;
160     cachedSizeHint = QSize(); // invalidate size hint
161     sizeHint();
162 }
```

7.62.3.37 QSize KDChart::TextLayoutItem::sizeHint () const `[virtual, inherited]`

pure virtual in QLayoutItem

Definition at line 295 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::sizeHintChanged().

Referenced by KDChart::TextLayoutItem::maximumSize(), KDChart::CartesianAxis::maximumSize(), KDChart::TextLayoutItem::minimumSize(), KDChart::CartesianAxis::paintCtx(), KDChart::TextLayoutItem::setAutoReferenceArea(), KDChart::TextLayoutItem::setText(), and KDChart::TextLayoutItem::setTextAttributes().

```
296 {
297     if( realFontWasRecalculated() )
298     {
299         const QSize newSizeHint( calcSizeHint( cachedFont ) );
300         if( newSizeHint != cachedSizeHint ){
301             cachedSizeHint = newSizeHint;
302             sizeHintChanged();
303         }
304     }
305     //qDebug() << "-------- KDChart::TextLayoutItem::sizeHint() returns:"<<cachedSizeHint<<" ---------
306     return cachedSizeHint;
307 }
```

7.62.3.38 void KDChart::AbstractLayoutItem::sizeHintChanged () const `[virtual, inherited]`

Report changed size hint: ask the parent widget to recalculate the layout.

Definition at line 86 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextLayoutItem::sizeHint().

```
87 {
88     // This is exactly like what QWidget::updateGeometry does.
89 //  qDebug("KDChart::AbstractLayoutItem::sizeHintChanged() called");
90     if( mParent ) {
91         if ( mParent->layout() )
92             mParent->layout()->invalidate();
93         else
94             QApplication::postEvent( mParent, new QEvent( QEvent::LayoutRequest ) );
95     }
96 }
```

7.62.3.39 QString KDChart::TextLayoutItem::text () const `[inherited]`

Definition at line 147 of file KDChartLayoutItems.cpp.

Referenced by KDChart::CartesianAxis::paintCtx().

```
148 {
149     return mText;
150 }
```

7.62.3.40 KDChart::TextAttributes KDChart::TextLayoutItem::textAttributes () const `[inherited]`

Returns the text attributes to be used for this item.

See also:
setTextAttributes

Definition at line 169 of file KDChartLayoutItems.cpp.

Referenced by KDChart::HeaderFooter::clone().

```
170 {
171     return mAttributes;
172 }
```

7.62.4 Member Data Documentation

7.62.4.1 Q_SIGNALS KDChart::TextArea::__pad0__ `[protected]`

Reimplemented in KDChart::HeaderFooter.

Definition at line 86 of file KDChartTextArea.h.

7.62.4.2 QWidget∗ KDChart::AbstractLayoutItem::mParent `[protected, inherited]`

Definition at line 88 of file KDChartLayoutItems.h.

Referenced by KDChart::AbstractLayoutItem::setParentWidget().

7.62.4.3 QLayout∗ KDChart::AbstractLayoutItem::mParentLayout `[protected, inherited]`

Definition at line 89 of file KDChartLayoutItems.h.

The documentation for this class was generated from the following files:

- KDChartTextArea.h
- KDChartTextArea.cpp

7.63 KDChart::TextAttributes Class Reference

`#include <KDChartTextAttributes.h>`

7.63.1 Detailed Description

A set of text attributes.

TextAttributes encapsulates settings that have to do with text. This includes font, fontsize, color, whether the text is rotated, etc

Definition at line 50 of file KDChartTextAttributes.h.

Public Member Functions

- bool autoRotate () const
- bool autoShrink () const
- const QFont calculatedFont (const QObject ∗autoReferenceArea, KDChartEnums::MeasureOrientation autoReferenceOrientation) const

  Returns the font in the size that is used at drawing time.

- const qreal calculatedFontSize (const QObject ∗autoReferenceArea, KDChartEnums::MeasureOrientation autoReferenceOrientation) const

  Returns the font size that is used at drawing time.

- QFont font () const
- Measure fontSize () const
- bool hasAbsoluteFontSize () const
- bool isVisible () const
- Measure minimalFontSize () const
- bool operator!= (const TextAttributes &other) const
- TextAttributes & operator= (const TextAttributes &)
- bool operator== (const TextAttributes &) const
- QPen pen () const
- int rotation () const
- void setAutoRotate (bool autoRotate)

  Set whether the text should be automatically rotated as needed when space is constraint.

- void setAutoShrink (bool autoShrink)

  Set whether the text should automatically be shrunk, if space is constraint.

- void setFont (const QFont &font)

  Set the font to be used for rendering the text.

- void setFontSize (const Measure &measure)

  Set the size of the font used for rendering text.

- void setMinimalFontSize (const Measure &measure)

  Set the minimal size of the font used for rendering text.

- void setPen (const QPen &pen)

  Set the pen to use for rendering the text.

- void setRotation (int rotation)

  Set the rotation angle to use for the text.

- void setVisible (bool visible)

  Set whether the text is to be rendered at all.

- TextAttributes (const TextAttributes &)
- TextAttributes ()
- ∼TextAttributes ()

7.63.2 Constructor & Destructor Documentation

7.63.2.1 KDChart::TextAttributes::TextAttributes ()

7.63.2.2 KDChart::TextAttributes::TextAttributes (const TextAttributes &)

7.63.2.3 KDChart::TextAttributes::∼TextAttributes ()

7.63.3 Member Function Documentation

7.63.3.1 bool KDChart::TextAttributes::autoRotate () const

Returns:

Whether text is automatically rotated when space is constrained.

Referenced by operator<<().

7.63.3.2 bool KDChart::TextAttributes::autoShrink () const

Returns:

Whether text is automatically shrunk if space is constraint.

Referenced by operator<<().

7.63.3.3 const QFont KDChart::TextAttributes::calculatedFont (const QObject ∗ *autoReferenceArea*, KDChartEnums::MeasureOrientation *autoReferenceOrientation*) const

Returns the font in the size that is used at drawing time.

This method is called at drawing time. It returns the font as it is used for rendering text, taking into account any measures that were set via setFontSize and/or setMinimalFontSize.

Referenced by KDChart::AbstractDiagram::paintDataValueText().

7.63.3.4 const qreal KDChart::TextAttributes::calculatedFontSize (const QObject ∗ *autoReferenceArea*, KDChartEnums::MeasureOrientation *autoReferenceOrientation*) const

Returns the font size that is used at drawing time.

This method is called at drawing time. It returns the font size as it is used for rendering text, taking into account any measures that were set via setFontSize and/or setMinimalFontSize.

Referenced by KDChart::Legend::buildLegend(), and KDChart::TextLayoutItem::realFontSize().

7.63.3.5 QFont KDChart::TextAttributes::font () const

Returns:

The font that is used for rendering text.

Referenced by operator<<().

7.63.3.6 Measure KDChart::TextAttributes::fontSize () const

Returns:

The measure used for the font size.

Referenced by KDChart::Chart::addLegend(), operator<<(), and KDChart::CartesianAxis::titleTextAttributes().

7.63.3.7 bool KDChart::TextAttributes::hasAbsoluteFontSize () const

Returns:

Whether the text has an absolute font size set.

7.63.3.8 bool KDChart::TextAttributes::isVisible () const

Returns:

Whether the text is visible.

Referenced by KDChart::Legend::buildLegend(), KDChart::CartesianAxis::maximumSize(), operator<<(), KDChart::CartesianAxis::paintCtx(), and KDChart::AbstractDiagram::paintDataValueText().

7.63.3.9 Measure KDChart::TextAttributes::minimalFontSize () const

Returns:

The measure used for the minimal font size.

Referenced by operator<<().

7.63.3.10 bool KDChart::TextAttributes::operator!= (const TextAttributes & *other*) const

Definition at line 57 of file KDChartTextAttributes.h.

```
58   { return !operator==(other); }
```

7.63.3.11 TextAttributes& KDChart::TextAttributes::operator= (const TextAttributes &)

7.63.3.12 bool KDChart::TextAttributes::operator== (const TextAttributes &) const

7.63.3.13 QPen KDChart::TextAttributes::pen () const

Returns:

The pen used for rendering the text.

Referenced by operator<<(), KDChart::TextLayoutItem::paint(), and KDChart::AbstractDiagram::paintDataValueText().

7.63.3.14 int KDChart::TextAttributes::rotation () const

Returns:

The rotation angle used for rendering the text.

Referenced by KDChart::TextLayoutItem::intersects(), operator<<(), KDChart::TextLayoutItem::paint(), and KDChart::AbstractDiagram::paintDataValueText().

7.63.3.15 void KDChart::TextAttributes::setAutoRotate (bool *autoRotate*)

Set whether the text should be automatically rotated as needed when space is constraint.

Parameters:

autoRotate Whether text should be automatically rotated.

7.63.3.16 void KDChart::TextAttributes::setAutoShrink (bool *autoShrink*)

Set whether the text should automatically be shrunk, if space is constraint.

Parameters:

autoShrink Whether text should be auto-shrunk.

7.63.3.17 void KDChart::TextAttributes::setFont (const QFont & *font*)

Set the font to be used for rendering the text.

Parameters:

font The font to use.

7.63.3.18 void KDChart::TextAttributes::setFontSize (const Measure & *measure*)

Set the size of the font used for rendering text.

Parameters:

measure The measure to use.

See also:

Measure

Referenced by KDChart::Chart::addLegend(), and KDChart::CartesianAxis::titleTextAttributes().

7.63.3.19 void KDChart::TextAttributes::setMinimalFontSize (const Measure & *measure*)

Set the minimal size of the font used for rendering text.

Parameters:

measure The measure to use.

See also:

Measure

7.63.3.20 void KDChart::TextAttributes::setPen (const QPen & *pen*)

Set the pen to use for rendering the text.

Parameters:

rotation The pen to use.

7.63.3.21 void KDChart::TextAttributes::setRotation (int *rotation*)

Set the rotation angle to use for the text.

Note:

For axis titles the rotation angle can be set to one of the following angles: 0, 90, 180, 270 Any other values specified will be replaced by the next smaller one of the allowed values, so no matter what you set the rotation will always be one of these four values.

Parameters:

rotation The rotation angle.

7.63.3.22 void KDChart::TextAttributes::setVisible (bool *visible*)

Set whether the text is to be rendered at all.

Parameters:

visible Whether the text is visible.

The documentation for this class was generated from the following file:

- KDChartTextAttributes.h

7.64 KDChart::TextLayoutItem Class Reference

`#include <KDChartLayoutItems.h>`

Inheritance diagram for KDChart::TextLayoutItem:Collaboration diagram for KDChart::TextLayoutItem:

Public Member Functions

- const QObject ∗ autoReferenceArea () const
- virtual Qt::Orientations expandingDirections () const

  pure virtual in QLayoutItem

- virtual QRect geometry () const

  pure virtual in QLayoutItem

- virtual bool intersects (const TextLayoutItem &other, const QPoint &myPos, const QPoint &otherPos) const
- virtual bool intersects (const TextLayoutItem &other, const QPointF &myPos, const QPointF &otherPos) const
- virtual bool isEmpty () const

  pure virtual in QLayoutItem

- virtual QSize maximumSize () const

  pure virtual in QLayoutItem

- virtual QSize minimumSize () const

  pure virtual in QLayoutItem

- virtual void paint (QPainter ∗)
- virtual void paintAll (QPainter &painter)

  Default impl: just call paint.

- virtual void paintCtx (PaintContext ∗context)

  Default impl: Paint the complete item using its layouted position and size.

- QLayout ∗ parentLayout ()
- virtual QFont realFont () const
- virtual qreal realFontSize () const
- void removeFromParentLayout ()
- void setAutoReferenceArea (const QObject ∗area)
- virtual void setGeometry (const QRect &r)

  pure virtual in QLayoutItem

- void setParentLayout (QLayout ∗lay)
- virtual void setParentWidget (QWidget ∗widget)

  Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

- void setText (const QString &text)
- void setTextAttributes (const TextAttributes &a)

Use this to specify the text attributes to be used for this item.

- virtual QSize sizeHint () const

  pure virtual in QLayoutItem

- virtual void sizeHintChanged () const

  Report changed size hint: ask the parent widget to recalculate the layout.

- QString text () const
- TextAttributes textAttributes () const

  Returns the text attributes to be used for this item.

- TextLayoutItem (const QString &text, const TextAttributes &attributes, const QObject ∗autoReferenceArea, KDChartEnums::MeasureOrientation autoReferenceOrientation, Qt::Alignment alignment=0)
- TextLayoutItem ()

Protected Attributes

- QWidget ∗ mParent
- QLayout ∗ mParentLayout

7.64.1 Constructor & Destructor Documentation

7.64.1.1 KDChart::TextLayoutItem::TextLayoutItem ()

Definition at line 115 of file KDChartLayoutItems.cpp.

```
    : AbstractLayoutItem( Qt::AlignLeft )
    , mText()
    , mAttributes()
    , mAutoReferenceArea( 0 )
    , mAutoReferenceOrientation( KDChartEnums::MeasureOrientationHorizontal )
    , cachedSizeHint() // default this to invalid to force just-in-time calculation before first use c
    , cachedFontSize( 0.0 )
    , cachedFont( mAttributes.font() )
{

}
```

7.64.1.2 KDChart::TextLayoutItem::TextLayoutItem (const QString & *text*, const TextAttributes & *attributes*, const QObject ∗ *autoReferenceArea*, KDChartEnums::MeasureOrientation *autoReferenceOrientation*, Qt::Alignment *alignment* = 0)

Definition at line 99 of file KDChartLayoutItems.cpp.

```
    : AbstractLayoutItem( alignment )
    , mText( text )
    , mAttributes( attributes )
    , mAutoReferenceArea( area )
    , mAutoReferenceOrientation( orientation )
    , cachedSizeHint() // default this to invalid to force just-in-time calculation before first use c
    , cachedFontSize( 0.0 )
    , cachedFont( mAttributes.font() )
{
}
```

7.64.2 Member Function Documentation

7.64.2.1 const QObject ∗ KDChart::TextLayoutItem::autoReferenceArea () const

Definition at line 135 of file KDChartLayoutItems.cpp.

Referenced by KDChart::HeaderFooter::setParent().

```
{
    return mAutoReferenceArea;
}
```

7.64.2.2 Qt::Orientations KDChart::TextLayoutItem::expandingDirections () const `[virtual]`

pure virtual in QLayoutItem

Definition at line 175 of file KDChartLayoutItems.cpp.

```
{
    return 0; // Grow neither vertically nor horizontally
}
```

7.64.2.3 QRect KDChart::TextLayoutItem::geometry () const `[virtual]`

pure virtual in QLayoutItem

Definition at line 180 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextArea::areaGeometry(), paint(), KDChart::TextArea::paintAll(), KDChart::CartesianAxis::paintCtx(), and KDChart::TextArea::paintIntoRect().

```
{
    return mRect;
}
```

7.64.2.4 bool KDChart::TextLayoutItem::intersects (const TextLayoutItem & *other*, const QPoint & *myPos*, const QPoint & *otherPos*) const `[virtual]`

Definition at line 254 of file KDChartLayoutItems.cpp.

References mAttributes, PI, rotatedCorners(), KDChart::TextAttributes::rotation(), and unrotatedSizeHint().

```
{
    if ( mAttributes.rotation() != other.mAttributes.rotation() )
    {
        // that's the code for the common case: the rotation angles don't need to match here
        QPolygon myPolygon(          rotatedCorners() );
        QPolygon otherPolygon( other.rotatedCorners() );

        // move the polygons to their positions
        myPolygon.translate( myPos );
        otherPolygon.translate( otherPos );

        // create regions out of it
        QRegion myRegion( myPolygon );
        QRegion otherRegion( otherPolygon );

        // now the question - do they intersect or not?
        return ! myRegion.intersect( otherRegion ).isEmpty();

    } else {
        // and that’s the code for the special case: the rotation angles match, which is less time cor
        const qreal angle = mAttributes.rotation() * PI / 180.0;
        // both sizes
        const QSizeF mySize(          unrotatedSizeHint() );
        const QSizeF otherSize( other.unrotatedSizeHint() );

        // that’s myP1 relative to myPos
        QPointF myP1( mySize.height() * sin( angle ), 0.0 );
        // that’s otherP1 to myPos
        QPointF otherP1 = QPointF( otherSize.height() * sin( angle ), 0.0 ) + otherPos - myPos;

        // now rotate both points the negative angle around myPos
        myP1 = QPointF( myP1.x() * cos( -angle ), myP1.x() * sin( -angle ) );
        qreal r = sqrt( otherP1.x() * otherP1.x() + otherP1.y() * otherP1.y() );
        otherP1 = QPointF( r * cos( -angle ), r * sin( -angle ) );

        // finally we look, whether both rectangles intersect or even not
        return QRectF( myP1, mySize ).intersects( QRectF( otherP1, otherSize ) );
    }
}
```

7.64.2.5 bool KDChart::TextLayoutItem::intersects (const TextLayoutItem & *other*, const QPointF & *myPos*, const QPointF & *otherPos*) const `[virtual]`

Definition at line 249 of file KDChartLayoutItems.cpp.

Referenced by KDChart::CartesianAxis::paintCtx().

```
{
    return intersects( other, myPos.toPoint(), otherPos.toPoint() );
}
```

7.64.2.6 bool KDChart::TextLayoutItem::isEmpty () const `[virtual]`

pure virtual in QLayoutItem

Definition at line 185 of file KDChartLayoutItems.cpp.

```
{
    return false; // never empty, otherwise the layout item would not exist
}
```

7.64.2.7 QSize KDChart::TextLayoutItem::maximumSize () const `[virtual]`

pure virtual in QLayoutItem

Definition at line 190 of file KDChartLayoutItems.cpp.

References sizeHint().

```
{
    return sizeHint(); // PENDING(kalle) Review, quite inflexible
}
```

7.64.2.8 QSize KDChart::TextLayoutItem::minimumSize () const `[virtual]`

pure virtual in QLayoutItem

Definition at line 195 of file KDChartLayoutItems.cpp.

References sizeHint().

```
{
    return sizeHint(); // PENDING(kalle) Review, quite inflexible
}
```

7.64.2.9 void KDChart::TextLayoutItem::paint (QPainter ∗) `[virtual]`

Implements KDChart::AbstractLayoutItem.

Definition at line 382 of file KDChartLayoutItems.cpp.

References geometry(), KDChart::TextAttributes::pen(), rotatedRect(), and KDChart::TextAttributes::rotation().

Referenced by KDChart::TextArea::paintAll(), and KDChart::CartesianAxis::paintCtx().

```
{
    // make sure, cached font is updated, if needed:
    // sizeHint();

    if( !mRect.isValid() )
        return;

    PainterSaver painterSaver( painter );
    painter->setFont( cachedFont );
    QRectF rect( geometry() );

// #ifdef DEBUG_ITEMS_PAINT
//     painter->setPen( Qt::black );
//     painter->drawRect( rect );
// #endif
    painter->translate( rect.center() );
    rect.moveTopLeft( QPointF( - rect.width() / 2, - rect.height() / 2 ) );
#ifdef DEBUG_ITEMS_PAINT
    painter->setPen( Qt::blue );
    painter->drawRect( rect );
#endif
    painter->rotate( mAttributes.rotation() );
    rect = rotatedRect( rect, mAttributes.rotation() );
#ifdef DEBUG_ITEMS_PAINT
    painter->setPen( Qt::red );
    painter->drawRect( rect );
#endif
    painter->setPen( mAttributes.pen() );
    painter->drawText( rect, Qt::AlignHCenter | Qt::AlignVCenter, mText );
//    if (  calcSizeHint( cachedFont ).width() > rect.width() )
//        qDebug() << "rect.width()" << rect.width() << "text.width()" << calcSizeHint( cachedFont ).w
//
//    //painter->drawText( rect, Qt::AlignHCenter | Qt::AlignVCenter, mText );
}
```

7.64.2.10 void KDChart::AbstractLayoutItem::paintAll (QPainter & *painter*) `[virtual, inherited]`

Default impl: just call paint.

Derived classes like KDChart::AbstractArea are providing additional action here.

Reimplemented in KDChart::AbstractArea, and KDChart::TextArea.

Definition at line 69 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint().

```
{
    paint( &painter );
}
```

7.64.2.11 void KDChart::AbstractLayoutItem::paintCtx (PaintContext ∗ *context*) `[virtual, inherited]`

Default impl: Paint the complete item using its layouted position and size.

Reimplemented in KDChart::CartesianAxis.

Definition at line 77 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint(), and KDChart::PaintContext::painter().

```
{
    if( context )
        paint( context->painter() );
}
```

7.64.2.12 QLayout∗ KDChart::AbstractLayoutItem::parentLayout () `[inherited]`

Definition at line 74 of file KDChartLayoutItems.h.

```
        {
            return mParentLayout;
        }
```

7.64.2.13 QFont KDChart::TextLayoutItem::realFont () const `[virtual]`

Definition at line 226 of file KDChartLayoutItems.cpp.

Referenced by KDChart::CartesianAxis::maximumSize(), and KDChart::CartesianAxis::paintCtx().

```
{
    realFontWasRecalculated(); // we can safely ignore the boolean return value
    return cachedFont;
}
```

7.64.2.14 qreal KDChart::TextLayoutItem::realFontSize () const `[virtual]`

Definition at line 206 of file KDChartLayoutItems.cpp.

References KDChart::TextAttributes::calculatedFontSize().

```
{
    return mAttributes.calculatedFontSize( mAutoReferenceArea, mAutoReferenceOrientation );
}
```

7.64.2.15 void KDChart::AbstractLayoutItem::removeFromParentLayout () `[inherited]`

Definition at line 78 of file KDChartLayoutItems.h.

Referenced by KDChart::Chart::takeCoordinatePlane().

```
       {
           if( mParentLayout ){
               if( widget() )
                   mParentLayout->removeWidget( widget() );
               else
                   mParentLayout->removeItem( this );
           }
       }
```

7.64.2.16 void KDChart::TextLayoutItem::setAutoReferenceArea (const QObject ∗ *area*)

Definition at line 128 of file KDChartLayoutItems.cpp.

References sizeHint().

Referenced by KDChart::HeaderFooter::setParent().

```
{
    mAutoReferenceArea = area;
    cachedSizeHint = QSize();
    sizeHint();
}
```

7.64.2.17 void KDChart::TextLayoutItem::setGeometry (const QRect & *r*) `[virtual]`

pure virtual in QLayoutItem

Definition at line 200 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextArea::paintAll(), KDChart::CartesianAxis::paintCtx(), and KDChart::TextArea::paintIntoRect().

```
{
    mRect = r;
}
```

7.64.2.18 void KDChart::AbstractLayoutItem::setParentLayout (QLayout ∗ *lay*) `[inherited]`

Definition at line 70 of file KDChartLayoutItems.h.

```
        {
            mParentLayout = lay;
        }
```

7.64.2.19 void KDChart::AbstractLayoutItem::setParentWidget (QWidget ∗ *widget*) `[virtual, inherited]`

Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

Thus, you need to call setParentWidget on every item, that has a non-fixed size.

Definition at line 64 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::mParent.

Referenced by KDChart::Legend::buildLegend(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
{
    mParent = widget;
}
```

7.64.2.20 void KDChart::TextLayoutItem::setText (const QString & *text*)

Definition at line 140 of file KDChartLayoutItems.cpp.

References sizeHint().

Referenced by KDChart::Widget::addHeaderFooter(), KDChart::CartesianAxis::maximumSize(), and KDChart::CartesianAxis::paintCtx().

```
{
    mText = text;
    cachedSizeHint = QSize();
    sizeHint();
}
```

7.64.2.21 void KDChart::TextLayoutItem::setTextAttributes (const TextAttributes & *a*)

Use this to specify the text attributes to be used for this item.

See also:

textAttributes

Definition at line 157 of file KDChartLayoutItems.cpp.

References sizeHint().

Referenced by KDChart::HeaderFooter::clone().

```
158 {
159     mAttributes = a;
160     cachedSizeHint = QSize(); // invalidate size hint
161     sizeHint();
162 }
```

7.64.2.22 QSize KDChart::TextLayoutItem::sizeHint () const `[virtual]`

pure virtual in QLayoutItem

Definition at line 295 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::sizeHintChanged().

Referenced by maximumSize(), KDChart::CartesianAxis::maximumSize(), minimumSize(), KDChart::CartesianAxis::paintCtx(), setAutoReferenceArea(), setText(), and setTextAttributes().

```
296 {
297     if( realFontWasRecalculated() )
298     {
299         const QSize newSizeHint( calcSizeHint( cachedFont ) );
300         if( newSizeHint != cachedSizeHint ){
301             cachedSizeHint = newSizeHint;
302             sizeHintChanged();
303         }
304     }
305     //qDebug() << "-------- KDChart::TextLayoutItem::sizeHint() returns:"<<cachedSizeHint<<" ---------
306     return cachedSizeHint;
307 }
```

7.64.2.23 void KDChart::AbstractLayoutItem::sizeHintChanged () const `[virtual, inherited]`

Report changed size hint: ask the parent widget to recalculate the layout.

Definition at line 86 of file KDChartLayoutItems.cpp.

Referenced by sizeHint().

```
87 {
88     // This is exactly like what QWidget::updateGeometry does.
89 //  qDebug("KDChart::AbstractLayoutItem::sizeHintChanged() called");
90     if( mParent ) {
91         if ( mParent->layout() )
92             mParent->layout()->invalidate();
93         else
94             QApplication::postEvent( mParent, new QEvent( QEvent::LayoutRequest ) );
95     }
96 }
```

7.64.2.24 QString KDChart::TextLayoutItem::text () const

Definition at line 147 of file KDChartLayoutItems.cpp.

Referenced by KDChart::CartesianAxis::paintCtx().

```
148 {
149     return mText;
150 }
```

7.64.2.25 KDChart::TextAttributes KDChart::TextLayoutItem::textAttributes () const

Returns the text attributes to be used for this item.

See also:
setTextAttributes

Definition at line 169 of file KDChartLayoutItems.cpp.

Referenced by KDChart::HeaderFooter::clone().

```
170 {
171     return mAttributes;
172 }
```

7.64.3 Member Data Documentation

7.64.3.1 QWidget∗ KDChart::AbstractLayoutItem::mParent `[protected, inherited]`

Definition at line 88 of file KDChartLayoutItems.h.

Referenced by KDChart::AbstractLayoutItem::setParentWidget().

7.64.3.2 QLayout∗ KDChart::AbstractLayoutItem::mParentLayout `[protected, inherited]`

Definition at line 89 of file KDChartLayoutItems.h.

The documentation for this class was generated from the following files:

- KDChartLayoutItems.h
- KDChartLayoutItems.cpp

7.65 KDChart::ThreeDBarAttributes Class Reference

`#include <KDChartThreeDBarAttributes.h>`

Inheritance diagram for KDChart::ThreeDBarAttributes:Collaboration diagram for KDChart::ThreeDBarAttributes:

Public Member Functions

- uint angle () const
- double depth () const
- bool isEnabled () const
- bool operator!= (const AbstractThreeDAttributes &other) const
- bool operator!= (const ThreeDBarAttributes &other) const
- ThreeDBarAttributes & operator= (const ThreeDBarAttributes &)
- bool operator== (const AbstractThreeDAttributes &) const
- bool operator== (const ThreeDBarAttributes &) const
- void setAngle (uint threeDAngle)
- void setDepth (double depth)
- void setEnabled (bool enabled)
- void setUseShadowColors (bool useShadowColors)
- ThreeDBarAttributes (const ThreeDBarAttributes &)
- ThreeDBarAttributes ()
- bool useShadowColors () const
- double validDepth () const
- ∼ThreeDBarAttributes ()

7.65.1 Constructor & Destructor Documentation

7.65.1.1 ThreeDBarAttributes::ThreeDBarAttributes ()

Definition at line 44 of file KDChartThreeDBarAttributes.cpp.

```
45     : AbstractThreeDAttributes( new Private() )
46 {
47
48 }
```

7.65.1.2 ThreeDBarAttributes::ThreeDBarAttributes (const ThreeDBarAttributes &)

Definition at line 50 of file KDChartThreeDBarAttributes.cpp.

References d.

```
51     : AbstractThreeDAttributes( new Private( *r.d) )
52 {
53 }
```

7.65.1.3 ThreeDBarAttributes::~ThreeDBarAttributes ()

Definition at line 65 of file KDChartThreeDBarAttributes.cpp.

```
66 {
67 }
```

7.65.2 Member Function Documentation

7.65.2.1 uint ThreeDBarAttributes::angle () const

Definition at line 98 of file KDChartThreeDBarAttributes.cpp.

References d.

Referenced by operator<<(), and operator==().

```
99 {
100     return d->angle;
101 }
```

7.65.2.2 double AbstractThreeDAttributes::depth () const `[inherited]`

Definition at line 103 of file KDChartAbstractThreeDAttributes.cpp.

References d.

Referenced by operator<<(), KDChart::AbstractThreeDAttributes::operator==(), KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), and KDChart::BarDiagram::paint().

```
104 {
105     return d->depth;
106 }
```

7.65.2.3 bool AbstractThreeDAttributes::isEnabled () const `[inherited]`

Definition at line 92 of file KDChartAbstractThreeDAttributes.cpp.

References d.

Referenced by operator<<(), KDChart::AbstractThreeDAttributes::operator==(), KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and KDChart::AbstractThreeDAttributes::validDepth().

```
93 {
94     return d->enabled;
95 }
```

7.65.2.4 bool KDChart::AbstractThreeDAttributes::operator!= (const AbstractThreeDAttributes & *other*) const `[inherited]`

Definition at line 57 of file KDChartAbstractThreeDAttributes.h.

```
57 { return !operator==(other); }
```

7.65.2.5 bool KDChart::ThreeDBarAttributes::operator!= (const ThreeDBarAttributes & *other*) const

Definition at line 53 of file KDChartThreeDBarAttributes.h.

```
53 { return !operator==(other); }
```

7.65.2.6 ThreeDBarAttributes & ThreeDBarAttributes::operator= (const ThreeDBarAttributes &)

Definition at line 55 of file KDChartThreeDBarAttributes.cpp.

References d.

```
56 {
57     if( this == &r )
58         return *this;
59
60     *d = *r.d;
61
62     return *this;
63 }
```

7.65.2.7 bool AbstractThreeDAttributes::operator== (const AbstractThreeDAttributes &) const `[inherited]`

Definition at line 72 of file KDChartAbstractThreeDAttributes.cpp.

References KDChart::AbstractThreeDAttributes::depth(), and KDChart::AbstractThreeDAttributes::isEnabled().

Referenced by KDChart::ThreeDPieAttributes::operator==(), KDChart::ThreeDLineAttributes::operator==(), and operator==().

```
73 {
74     if( isEnabled() == r.isEnabled() &&
75         depth() == r.depth() )
76         return true;
77     else
78         return false;
79 }
```

7.65.2.8 bool ThreeDBarAttributes::operator== (const ThreeDBarAttributes &) const

Definition at line 74 of file KDChartThreeDBarAttributes.cpp.

References angle(), KDChart::AbstractThreeDAttributes::operator==(), and useShadowColors().

```
75 {
76     return ( useShadowColors() == r.useShadowColors() &&
77              angle() == r.angle() &&
78              AbstractThreeDAttributes::operator==(r));
79 }
```

7.65.2.9 void ThreeDBarAttributes::setAngle (uint *threeDAngle*)

Definition at line 93 of file KDChartThreeDBarAttributes.cpp.

References d.

```
{
    d->angle = threeDAngle;
}
```

7.65.2.10 void AbstractThreeDAttributes::setDepth (double *depth*) `[inherited]`

Definition at line 97 of file KDChartAbstractThreeDAttributes.cpp.

References d.

```
{
    d->depth = depth;
}
```

7.65.2.11 void AbstractThreeDAttributes::setEnabled (bool *enabled*) `[inherited]`

Definition at line 87 of file KDChartAbstractThreeDAttributes.cpp.

References d.

```
{
    d->enabled = enabled;
}
```

7.65.2.12 void ThreeDBarAttributes::setUseShadowColors (bool *useShadowColors*)

Definition at line 83 of file KDChartThreeDBarAttributes.cpp.

References d.

```
{
    d->useShadowColors = shadowColors;
}
```

7.65.2.13 bool ThreeDBarAttributes::useShadowColors () const

Definition at line 88 of file KDChartThreeDBarAttributes.cpp.

References d.

Referenced by operator<<(), and operator==().

```
{
    return d->useShadowColors;
}
```

7.65.2.14 double AbstractThreeDAttributes::validDepth () const `[inherited]`

Definition at line 109 of file KDChartAbstractThreeDAttributes.cpp.

References d, and KDChart::AbstractThreeDAttributes::isEnabled().

Referenced by KDChart::LineDiagram::threeDItemDepth(), and KDChart::BarDiagram::threeDItemDepth().

```
110 {
111     return isEnabled() ? d->depth : 0.0;
112 }
```

The documentation for this class was generated from the following files:

- KDChartThreeDBarAttributes.h
- KDChartThreeDBarAttributes.cpp

7.66 KDChart::ThreeDLineAttributes Class Reference

`#include <KDChartThreeDLineAttributes.h>`

Inheritance diagram for KDChart::ThreeDLineAttributes:Collaboration diagram for KDChart::ThreeDLineAttributes:

Public Member Functions

- double depth () const
- bool isEnabled () const
- uint lineXRotation () const
- uint lineYRotation () const
- bool operator!= (const AbstractThreeDAttributes &other) const
- bool operator!= (const ThreeDLineAttributes &other) const
- ThreeDLineAttributes & operator= (const ThreeDLineAttributes &)
- bool operator== (const AbstractThreeDAttributes &) const
- bool operator== (const ThreeDLineAttributes &) const
- void setDepth (double depth)
- void setEnabled (bool enabled)
- void setLineXRotation (const uint degrees)
- void setLineYRotation (const uint degrees)
- ThreeDLineAttributes (const ThreeDLineAttributes &)
- ThreeDLineAttributes ()
- double validDepth () const
- ∼ThreeDLineAttributes ()

7.66.1 Constructor & Destructor Documentation

7.66.1.1 ThreeDLineAttributes::ThreeDLineAttributes ()

Definition at line 44 of file KDChartThreeDLineAttributes.cpp.

```
    : AbstractThreeDAttributes( new Private() )
{

}
```

7.66.1.2 ThreeDLineAttributes::ThreeDLineAttributes (const ThreeDLineAttributes &)

Definition at line 50 of file KDChartThreeDLineAttributes.cpp.

References d.

```
    : AbstractThreeDAttributes( new Private( *r.d) )
{
}
```

7.66.1.3 ThreeDLineAttributes::∼ThreeDLineAttributes ()

Definition at line 65 of file KDChartThreeDLineAttributes.cpp.

```
66 {
67 }
```

7.66.2 Member Function Documentation

7.66.2.1 double AbstractThreeDAttributes::depth () const `[inherited]`

Definition at line 103 of file KDChartAbstractThreeDAttributes.cpp.

References d.

Referenced by operator<<(), KDChart::AbstractThreeDAttributes::operator==(), KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), and KDChart::BarDiagram::paint().

```
104 {
105     return d->depth;
106 }
```

7.66.2.2 bool AbstractThreeDAttributes::isEnabled () const `[inherited]`

Definition at line 92 of file KDChartAbstractThreeDAttributes.cpp.

References d.

Referenced by operator<<(), KDChart::AbstractThreeDAttributes::operator==(), KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and KDChart::AbstractThreeDAttributes::validDepth().

```
93 {
94     return d->enabled;
95 }
```

7.66.2.3 uint ThreeDLineAttributes::lineXRotation () const

Definition at line 88 of file KDChartThreeDLineAttributes.cpp.

References d.

Referenced by operator<<(), and operator==().

```
89 {
90     return d->lineXRotation;
91 }
```

7.66.2.4 uint ThreeDLineAttributes::lineYRotation () const

Definition at line 98 of file KDChartThreeDLineAttributes.cpp.

References d.

Referenced by operator<<(), and operator==().

```
99 {
100     return d->lineYRotation;
101 }
```

7.66.2.5 bool KDChart::AbstractThreeDAttributes::operator!= (const AbstractThreeDAttributes & *other*) const `[inherited]`

Definition at line 57 of file KDChartAbstractThreeDAttributes.h.

```
57 { return !operator==(other); }
```

7.66.2.6 bool KDChart::ThreeDLineAttributes::operator!= (const ThreeDLineAttributes & *other*) const

Definition at line 51 of file KDChartThreeDLineAttributes.h.

```
51 { return !operator==(other); }
```

7.66.2.7 ThreeDLineAttributes & ThreeDLineAttributes::operator= (const ThreeDLineAttributes &)

Definition at line 55 of file KDChartThreeDLineAttributes.cpp.

References d.

```
56 {
57     if( this == &r )
58         return *this;
59 
60     *d = *r.d;
61 
62     return *this;
63 }
```

7.66.2.8 bool AbstractThreeDAttributes::operator== (const AbstractThreeDAttributes &) const `[inherited]`

Definition at line 72 of file KDChartAbstractThreeDAttributes.cpp.

References KDChart::AbstractThreeDAttributes::depth(), and KDChart::AbstractThreeDAttributes::isEnabled().

Referenced by KDChart::ThreeDPieAttributes::operator==(), operator==(), and KDChart::ThreeDBarAttributes::operator==().

```
73 {
74     if( isEnabled() == r.isEnabled() &&
75         depth() == r.depth() )
76         return true;
77     else
78         return false;
79 }
```

7.66.2.9 bool ThreeDLineAttributes::operator== (const ThreeDLineAttributes &) const

Definition at line 74 of file KDChartThreeDLineAttributes.cpp.

References lineXRotation(), lineYRotation(), and KDChart::AbstractThreeDAttributes::operator==().

```
{
    return ( lineXRotation() == r.lineXRotation() &&
             lineYRotation() == r.lineYRotation() &&
             AbstractThreeDAttributes::operator==(r));
}
```

7.66.2.10 void AbstractThreeDAttributes::setDepth (double *depth*) `[inherited]`

Definition at line 97 of file KDChartAbstractThreeDAttributes.cpp.

References d.

```
{
    d->depth = depth;
}
```

7.66.2.11 void AbstractThreeDAttributes::setEnabled (bool *enabled*) `[inherited]`

Definition at line 87 of file KDChartAbstractThreeDAttributes.cpp.

References d.

```
{
    d->enabled = enabled;
}
```

7.66.2.12 void ThreeDLineAttributes::setLineXRotation (const uint *degrees*)

Definition at line 83 of file KDChartThreeDLineAttributes.cpp.

References d.

```
{
    d->lineXRotation = degrees;
}
```

7.66.2.13 void ThreeDLineAttributes::setLineYRotation (const uint *degrees*)

Definition at line 93 of file KDChartThreeDLineAttributes.cpp.

References d.

```
{
    d->lineYRotation = degrees;
}
```

7.66.2.14 double AbstractThreeDAttributes::validDepth () const `[inherited]`

Definition at line 109 of file KDChartAbstractThreeDAttributes.cpp.

References d, and KDChart::AbstractThreeDAttributes::isEnabled().

Referenced by KDChart::LineDiagram::threeDItemDepth(), and KDChart::BarDiagram::threeDItemDepth().

```
110 {
111     return isEnabled() ? d->depth : 0.0;
112 }
```

The documentation for this class was generated from the following files:

- KDChartThreeDLineAttributes.h
- KDChartThreeDLineAttributes.cpp

7.67 KDChart::ThreeDPieAttributes Class Reference

`#include <KDChartThreeDPieAttributes.h>`

Inheritance diagram for KDChart::ThreeDPieAttributes:Collaboration diagram for KDChart::ThreeDPieAttributes:

Public Member Functions

- double depth () const
- bool isEnabled () const
- bool operator!= (const AbstractThreeDAttributes &other) const
- bool operator!= (const ThreeDPieAttributes &other) const
- ThreeDPieAttributes & operator= (const ThreeDPieAttributes &)
- bool operator== (const AbstractThreeDAttributes &) const
- bool operator== (const ThreeDPieAttributes &) const
- void setDepth (double depth)
- void setEnabled (bool enabled)
- void setUseShadowColors (bool useShadowColors)
- ThreeDPieAttributes (const ThreeDPieAttributes &)
- ThreeDPieAttributes ()
- bool useShadowColors () const
- double validDepth () const
- ∼ThreeDPieAttributes ()

7.67.1 Constructor & Destructor Documentation

7.67.1.1 ThreeDPieAttributes::ThreeDPieAttributes ()

Definition at line 43 of file KDChartThreeDPieAttributes.cpp.

```
44     : AbstractThreeDAttributes( new Private() )
45 {
46
47 }
```

7.67.1.2 ThreeDPieAttributes::ThreeDPieAttributes (const ThreeDPieAttributes &)

Definition at line 49 of file KDChartThreeDPieAttributes.cpp.

References d.

```
50     : AbstractThreeDAttributes( new Private( *r.d) )
51 {
52 }
```

7.67.1.3 ThreeDPieAttributes::∼ThreeDPieAttributes ()

Definition at line 64 of file KDChartThreeDPieAttributes.cpp.

```
65 {
66 }
```

7.67.2 Member Function Documentation

7.67.2.1 double AbstractThreeDAttributes::depth () const `[inherited]`

Definition at line 103 of file KDChartAbstractThreeDAttributes.cpp.

References d.

Referenced by operator<<(), KDChart::AbstractThreeDAttributes::operator==(), KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), and KDChart::BarDiagram::paint().

```
104 {
105     return d->depth;
106 }
```

7.67.2.2 bool AbstractThreeDAttributes::isEnabled () const `[inherited]`

Definition at line 92 of file KDChartAbstractThreeDAttributes.cpp.

References d.

Referenced by operator<<(), KDChart::AbstractThreeDAttributes::operator==(), KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::BarDiagram::paint(), and KDChart::AbstractThreeDAttributes::validDepth().

```
93 {
94     return d->enabled;
95 }
```

7.67.2.3 bool KDChart::AbstractThreeDAttributes::operator!= (const AbstractThreeDAttributes & *other*) const `[inherited]`

Definition at line 57 of file KDChartAbstractThreeDAttributes.h.

```
57 { return !operator==(other); }
```

7.67.2.4 bool KDChart::ThreeDPieAttributes::operator!= (const ThreeDPieAttributes & *other*) const

Definition at line 49 of file KDChartThreeDPieAttributes.h.

```
49 { return !operator==(other); }
```

7.67.2.5 ThreeDPieAttributes & ThreeDPieAttributes::operator= (const ThreeDPieAttributes &)

Definition at line 54 of file KDChartThreeDPieAttributes.cpp.

References d.

```
{
    if( this == &r )
        return *this;

    *d = *r.d;

    return *this;
}
```

7.67.2.6 bool AbstractThreeDAttributes::operator== (const AbstractThreeDAttributes &) const `[inherited]`

Definition at line 72 of file KDChartAbstractThreeDAttributes.cpp.

References KDChart::AbstractThreeDAttributes::depth(), and KDChart::AbstractThreeDAttributes::isEnabled().

Referenced by operator==(), KDChart::ThreeDLineAttributes::operator==(), and KDChart::ThreeDBarAttributes::operator==().

```
{
    if( isEnabled() == r.isEnabled() &&
        depth() == r.depth() )
        return true;
    else
        return false;
}
```

7.67.2.7 bool ThreeDPieAttributes::operator== (const ThreeDPieAttributes &) const

Definition at line 73 of file KDChartThreeDPieAttributes.cpp.

References KDChart::AbstractThreeDAttributes::operator==(), and useShadowColors().

```
{
    return ( useShadowColors() == r.useShadowColors() &&
             AbstractThreeDAttributes::operator==(r));
}
```

7.67.2.8 void AbstractThreeDAttributes::setDepth (double *depth*) `[inherited]`

Definition at line 97 of file KDChartAbstractThreeDAttributes.cpp.

References d.

```
{
    d->depth = depth;
}
```

7.67.2.9 void AbstractThreeDAttributes::setEnabled (bool *enabled*) `[inherited]`

Definition at line 87 of file KDChartAbstractThreeDAttributes.cpp.

References d.

```
88 {
89     d->enabled = enabled;
90 }
```

7.67.2.10 void ThreeDPieAttributes::setUseShadowColors (bool *useShadowColors*)

Definition at line 81 of file KDChartThreeDPieAttributes.cpp.

References d.

```
82 {
83     d->useShadowColors = shadowColors;
84 }
```

7.67.2.11 bool ThreeDPieAttributes::useShadowColors () const

Definition at line 86 of file KDChartThreeDPieAttributes.cpp.

References d.

Referenced by operator<<(), and operator==().

```
87 {
88     return d->useShadowColors;
89 }
```

7.67.2.12 double AbstractThreeDAttributes::validDepth () const `[inherited]`

Definition at line 109 of file KDChartAbstractThreeDAttributes.cpp.

References d, and KDChart::AbstractThreeDAttributes::isEnabled().

Referenced by KDChart::LineDiagram::threeDItemDepth(), and KDChart::BarDiagram::threeDItemDepth().

```
110 {
111     return isEnabled() ? d->depth : 0.0;
112 }
```

The documentation for this class was generated from the following files:

- KDChartThreeDPieAttributes.h
- KDChartThreeDPieAttributes.cpp

7.68 KDChart::VerticalLineLayoutItem Class Reference

`#include <KDChartLayoutItems.h>`

Inheritance diagram for KDChart::VerticalLineLayoutItem:Collaboration diagram for KDChart::VerticalLineLayoutItem:

Public Member Functions

- virtual Qt::Orientations expandingDirections () const
- virtual QRect geometry () const
- virtual bool isEmpty () const
- virtual QSize maximumSize () const
- virtual QSize minimumSize () const
- virtual void paint (QPainter ∗)
- virtual void paintAll (QPainter &painter)

  Default impl: just call paint.

- virtual void paintCtx (PaintContext ∗context)

  Default impl: Paint the complete item using its layouted position and size.

- QLayout ∗ parentLayout ()
- void removeFromParentLayout ()
- virtual void setGeometry (const QRect &r)
- void setParentLayout (QLayout ∗lay)
- virtual void setParentWidget (QWidget ∗widget)

  Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

- virtual QSize sizeHint () const
- virtual void sizeHintChanged () const

  Report changed size hint: ask the parent widget to recalculate the layout.

- VerticalLineLayoutItem ()

Protected Attributes

- QWidget ∗ mParent
- QLayout ∗ mParentLayout

7.68.1 Constructor & Destructor Documentation

7.68.1.1 KDChart::VerticalLineLayoutItem::VerticalLineLayoutItem ()

Definition at line 469 of file KDChartLayoutItems.cpp.

```
470     : AbstractLayoutItem( Qt::AlignCenter )
471 {
472 }
```

7.68.2 Member Function Documentation

7.68.2.1 Qt::Orientations KDChart::VerticalLineLayoutItem::expandingDirections () const `[virtual]`

Definition at line 474 of file KDChartLayoutItems.cpp.

```
{
    return Qt::Vertical|Qt::Vertical; // Grow both vertically, and horizontally
}
```

7.68.2.2 QRect KDChart::VerticalLineLayoutItem::geometry () const `[virtual]`

Definition at line 479 of file KDChartLayoutItems.cpp.

```
{
    return mRect;
}
```

7.68.2.3 bool KDChart::VerticalLineLayoutItem::isEmpty () const `[virtual]`

Definition at line 484 of file KDChartLayoutItems.cpp.

```
{
    return false; // never empty, otherwise the layout item would not exist
}
```

7.68.2.4 QSize KDChart::VerticalLineLayoutItem::maximumSize () const `[virtual]`

Definition at line 489 of file KDChartLayoutItems.cpp.

```
{
    return QSize( QWIDGETSIZE_MAX, QWIDGETSIZE_MAX );
}
```

7.68.2.5 QSize KDChart::VerticalLineLayoutItem::minimumSize () const `[virtual]`

Definition at line 494 of file KDChartLayoutItems.cpp.

```
{
    return QSize( 0, 0 );
}
```

7.68.2.6 void KDChart::VerticalLineLayoutItem::paint (QPainter $*$) `[virtual]`

Implements KDChart::AbstractLayoutItem.

Definition at line 510 of file KDChartLayoutItems.cpp.

```
511 {
512     if( !mRect.isValid() )
513         return;
514
515     painter->drawLine( QPointF( mRect.center().x(), mRect.top() ),
516                        QPointF( mRect.center().x(), mRect.bottom() ) );
517 }
```

7.68.2.7 void KDChart::AbstractLayoutItem::paintAll (QPainter & *painter*) `[virtual, inherited]`

Default impl: just call paint.

Derived classes like KDChart::AbstractArea are providing additional action here.

Reimplemented in KDChart::AbstractArea, and KDChart::TextArea.

Definition at line 69 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint().

```
70 {
71     paint( &painter );
72 }
```

7.68.2.8 void KDChart::AbstractLayoutItem::paintCtx (PaintContext ∗ *context*) `[virtual, inherited]`

Default impl: Paint the complete item using its layouted position and size.

Reimplemented in KDChart::CartesianAxis.

Definition at line 77 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::paint(), and KDChart::PaintContext::painter().

```
78 {
79     if( context )
80         paint( context->painter() );
81 }
```

7.68.2.9 QLayout∗ KDChart::AbstractLayoutItem::parentLayout () `[inherited]`

Definition at line 74 of file KDChartLayoutItems.h.

```
75         {
76             return mParentLayout;
77         }
```

7.68.2.10 void KDChart::AbstractLayoutItem::removeFromParentLayout () `[inherited]`

Definition at line 78 of file KDChartLayoutItems.h.

Referenced by KDChart::Chart::takeCoordinatePlane().

```
79         {
80             if( mParentLayout ){
81                 if( widget() )
82                     mParentLayout->removeWidget( widget() );
83                 else
84                     mParentLayout->removeItem( this );
85             }
86         }
```

7.68.2.11 void KDChart::VerticalLineLayoutItem::setGeometry (const QRect & *r*) `[virtual]`

Definition at line 499 of file KDChartLayoutItems.cpp.

```
500 {
501     mRect = r;
502 }
```

7.68.2.12 void KDChart::AbstractLayoutItem::setParentLayout (QLayout ∗ *lay*) `[inherited]`

Definition at line 70 of file KDChartLayoutItems.h.

```
71         {
72             mParentLayout = lay;
73         }
```

7.68.2.13 void KDChart::AbstractLayoutItem::setParentWidget (QWidget ∗ *widget*) `[virtual, inherited]`

Inform the item about its widget: This enables the item, to trigger that widget's update, whenever the size of the item's contents has changed.

Thus, you need to call setParentWidget on every item, that has a non-fixed size.

Definition at line 64 of file KDChartLayoutItems.cpp.

References KDChart::AbstractLayoutItem::mParent.

Referenced by KDChart::Legend::buildLegend(), and KDChart::AbstractCartesianDiagram::takeAxis().

```
65 {
66     mParent = widget;
67 }
```

7.68.2.14 QSize KDChart::VerticalLineLayoutItem::sizeHint () const `[virtual]`

Definition at line 504 of file KDChartLayoutItems.cpp.

```
505 {
506     return QSize( 3, -1 ); // see qframe.cpp
507 }
```

7.68.2.15 void KDChart::AbstractLayoutItem::sizeHintChanged () const `[virtual, inherited]`

Report changed size hint: ask the parent widget to recalculate the layout.

Definition at line 86 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextLayoutItem::sizeHint().

```
{
    // This is exactly like what QWidget::updateGeometry does.
//  qDebug("KDChart::AbstractLayoutItem::sizeHintChanged() called");
    if( mParent ) {
        if ( mParent->layout() )
            mParent->layout()->invalidate();
        else
            QApplication::postEvent( mParent, new QEvent( QEvent::LayoutRequest ) );
    }
}
```

7.68.3 Member Data Documentation

7.68.3.1 QWidget∗ KDChart::AbstractLayoutItem::mParent `[protected, inherited]`

Definition at line 88 of file KDChartLayoutItems.h.

Referenced by KDChart::AbstractLayoutItem::setParentWidget().

7.68.3.2 QLayout∗ KDChart::AbstractLayoutItem::mParentLayout `[protected, inherited]`

Definition at line 89 of file KDChartLayoutItems.h.

The documentation for this class was generated from the following files:

- KDChartLayoutItems.h
- KDChartLayoutItems.cpp

7.69 KDChart::Widget Class Reference

`#include <KDChartWidget.h>`

Inheritance diagram for KDChart::Widget:Collaboration diagram for KDChart::Widget:

7.69.1 Detailed Description

The KD Chart widget for usage without Model/View.

If you want to use KD Chart with Model/View, use KDChart::Chart instead.

Definition at line 63 of file KDChartWidget.h.

Public Types

- enum ChartType {
  NoType,
  Bar,
  Line,
  Pie,
  Ring,
  Polar }
- enum SubType {
  Normal,
  Stacked,
  Percent,
  Rows }

  Sub type values, matching the values defines for the respective Diagram classes.

Public Member Functions

- void addHeaderFooter (HeaderFooter ∗header)

  Adds an existing header / footer object.

- void addHeaderFooter (const QString &text, HeaderFooter::HeaderFooterType type, Position position)

  Adds a new header/footer with the given text to the position.

- void addLegend (Legend ∗legend)

  Adds a new, already existing, legend.

- void addLegend (Position position)

  Adds an empty legend on the given position.

- QList< HeaderFooter ∗ > allHeadersFooters ()

  Returns a list with all headers.

- QList< Legend ∗ > allLegends ()

  Returns a list with all legends.

- BarDiagram ∗ barDiagram ()

  If the current diagram is a BarDiagram, it is returnd; otherwise 0 is returned.

- AbstractCoordinatePlane ∗ coordinatePlane ()

  Returns a pointer to the current coordinate plane.

- AbstractDiagram ∗ diagram ()

  Returns a pointer to the current diagram.

- HeaderFooter ∗ firstHeaderFooter ()

  Returns the first of all headers.

- int globalLeadingBottom () const

  Returns the bottom leading (border).

- int globalLeadingLeft () const

  Returns the left leading (border).

- int globalLeadingRight () const

  Returns the right leading (border).

- int globalLeadingTop () const

  Returns the top leading (border).

- Legend ∗ legend ()

  Returns the first of all legends.

- LineDiagram ∗ lineDiagram ()

  If the current diagram is a LineDiagram, it is returnd; otherwise 0 is returned.

- PieDiagram ∗ pieDiagram ()

  If the current diagram is a PieDiagram, it is returnd; otherwise 0 is returned.

- PolarDiagram ∗ polarDiagram ()

  If the current diagram is a PolarDiagram, it is returnd; otherwise 0 is returned.

- void replaceHeaderFooter (HeaderFooter ∗header, HeaderFooter ∗oldHeader=0)

  Replaces the old header (or footer, resp.), or appends the new header or footer, it there is none yet.

- void replaceLegend (Legend ∗legend, Legend ∗oldLegend=0)
- void resetData ()

  Resets all data.

- RingDiagram ∗ ringDiagram ()

  If the current diagram is a RingDiagram, it is returnd; otherwise 0 is returned.

- void setDataCell (int row, int column, QPair< double, double > data)

  Sets the data for a given column using an (X, Y) QPair of doubles.

- void setDataCell (int row, int column, double data)

  Sets the Y value data for a given cell.

- void setDataset (int column, const QVector< QPair< double, double > > &data, const QString &title=QString())

  Sets the data in the given column using a QVector of QPairs of double for the (X, Y) values.

- void setDataset (int column, const QVector< double > &data, const QString &title=QString())

  Sets the data in the given column using a QVector of double for the Y values.

- void setGlobalLeadingBottom (int leading)

  Sets the bottom leading (border).

- void setGlobalLeadingLeft (int leading)

  Sets the left leading (border).

- void setGlobalLeadingRight (int leading)

  Sets the right leading (border).

- void setGlobalLeadingTop (int leading)

  Sets the top leading (border).

- void setSubType (SubType subType)

  Sets the type of the chart without changing the main type.

- SubType subType () const

  Returns the sub-type of the chart.

- void takeHeaderFooter (HeaderFooter ∗header)

  Remove the header (or footer, resp.) from the widget, without deleting it.

- void takeLegend (Legend ∗legend)
- ChartType type () const

  Returns the type of the chart.

- Widget (QWidget ∗parent=0)

  Standard Qt-style Constructor.

- ∼Widget ()

  Destructor.

Public Attributes

- public int int int bottom
- public Q_SLOTS: void setGlobalLeading(int left
- public int int right
- public SubType subType = Normal)
- public int top

7.69.2 Member Enumeration Documentation

7.69.2.1 enum KDChart::Widget::ChartType

Enumeration values:

NoType

Bar

Line

Pie

Ring

Polar

Definition at line 203 of file KDChartWidget.h.

```
203 { NoType, Bar, Line, Pie, Ring, Polar };
```

7.69.2.2 enum KDChart::Widget::SubType

Sub type values, matching the values defines for the respective Diagram classes.

Enumeration values:

Normal

Stacked

Percent

Rows

Definition at line 209 of file KDChartWidget.h.

```
209 { Normal, Stacked, Percent, Rows };
```

7.69.3 Constructor & Destructor Documentation

7.69.3.1 Widget::Widget (QWidget ∗ *parent* = 0) `[explicit]`

Standard Qt-style Constructor.

Creates a new widget with all data initialized empty.

Parameters:

parent the widget parent; passed on to QWidget

Definition at line 82 of file KDChartWidget.cpp.

References Line.

```
82                                    :
83     QWidget(parent), _d( new Private( this ) )
84 {
85     // as default we have a cartesian coordinate plane ...
86     // ... and a line diagram
87     setType( Line );
88 }
```

7.69.3.2 Widget::∼Widget ()

Destructor.

Definition at line 93 of file KDChartWidget.cpp.

```
94 {
95     delete _d; _d = 0;
96 }
```

7.69.4 Member Function Documentation

7.69.4.1 void Widget::addHeaderFooter (HeaderFooter ∗ *header*)

Adds an existing header / footer object.

See also:
replaceHeaderFooter, takeHeaderFooter

Definition at line 286 of file KDChartWidget.cpp.

References d, and KDChart::HeaderFooter::setParent().

```
287 {
288     header->setParent( &d->m_chart );
289     d->m_chart.addHeaderFooter( header ); // we need this explicit call !
290 }
```

7.69.4.2 void Widget::addHeaderFooter (const QString & *text*, HeaderFooter::HeaderFooterType *type*, Position *position*)

Adds a new header/footer with the given text to the position.

Definition at line 272 of file KDChartWidget.cpp.

References d, KDChart::HeaderFooter::setPosition(), KDChart::TextLayoutItem::setText(), and KDChart::HeaderFooter::setType().

```
275 {
276     HeaderFooter* newHeader = new HeaderFooter( &d->m_chart );
277     newHeader->setType( type );
278     newHeader->setPosition( position );
279     newHeader->setText( text );
280     d->m_chart.addHeaderFooter( newHeader ); // we need this explicit call !
281 }
```

7.69.4.3 void Widget::addLegend (Legend ∗ *legend*)

Adds a new, already existing, legend.

Definition at line 332 of file KDChartWidget.cpp.

References d, diagram(), and KDChart::Legend::setDiagram().

```
{
    legend->setDiagram( diagram() );
    legend->setParent( &d->m_chart );
    d->m_chart.addLegend( legend );
}
```

7.69.4.4 void Widget::addLegend (Position *position*)

Adds an empty legend on the given position.

Definition at line 322 of file KDChartWidget.cpp.

References d, diagram(), and KDChart::Legend::setPosition().

```
{
    Legend* legend = new Legend( diagram(), &d->m_chart );
    legend->setPosition( position );
    d->m_chart.addLegend( legend );
}
```

7.69.4.5 QList< KDChart::HeaderFooter ∗ > Widget::allHeadersFooters ()

Returns a list with all headers.

Definition at line 264 of file KDChartWidget.cpp.

References d.

```
{
    return d->m_chart.headerFooters();
}
```

7.69.4.6 QList< KDChart::Legend ∗ > Widget::allLegends ()

Returns a list with all legends.

Definition at line 314 of file KDChartWidget.cpp.

References d.

```
{
    return d->m_chart.legends();
}
```

7.69.4.7 BarDiagram ∗ Widget::barDiagram ()

If the current diagram is a BarDiagram, it is returnd; otherwise 0 is returned.

This function provides type-safe casting.

Definition at line 359 of file KDChartWidget.cpp.

References diagram().

```
{
    return dynamic_cast<BarDiagram*>( diagram() );
}
```

7.69.4.8 AbstractCoordinatePlane ∗ Widget::coordinatePlane ()

Returns a pointer to the current coordinate plane.

Definition at line 380 of file KDChartWidget.cpp.

References d.

Referenced by diagram().

```
{
    return d->m_chart.coordinatePlane();
}
```

7.69.4.9 AbstractDiagram ∗ Widget::diagram ()

Returns a pointer to the current diagram.

Definition at line 351 of file KDChartWidget.cpp.

References coordinatePlane(), and KDChart::AbstractCoordinatePlane::diagram().

Referenced by addLegend(), barDiagram(), lineDiagram(), pieDiagram(), polarDiagram(), replaceLegend(), ringDiagram(), and setSubType().

```
{
    if ( coordinatePlane() == 0 )
        qDebug() << "diagram(): coordinatePlane() was NULL";

    return coordinatePlane()->diagram();
}
```

7.69.4.10 KDChart::HeaderFooter ∗ Widget::firstHeaderFooter ()

Returns the first of all headers.

Definition at line 256 of file KDChartWidget.cpp.

References d.

```
{
    return d->m_chart.headerFooter();
}
```

7.69.4.11 int Widget::globalLeadingBottom () const

Returns the bottom leading (border).

Definition at line 248 of file KDChartWidget.cpp.

References d.

```
{
    return d->m_chart.globalLeadingBottom();
}
```

7.69.4.12 int Widget::globalLeadingLeft () const

Returns the left leading (border).

Definition at line 200 of file KDChartWidget.cpp.

References d.

```
201 {
202     return d->m_chart.globalLeadingLeft();
203 }
```

7.69.4.13 int Widget::globalLeadingRight () const

Returns the right leading (border).

Definition at line 232 of file KDChartWidget.cpp.

References d.

```
233 {
234     return d->m_chart.globalLeadingRight();
235 }
```

7.69.4.14 int Widget::globalLeadingTop () const

Returns the top leading (border).

Definition at line 216 of file KDChartWidget.cpp.

References d.

```
217 {
218     return d->m_chart.globalLeadingTop();
219 }
```

7.69.4.15 KDChart::Legend ∗ Widget::legend ()

Returns the first of all legends.

Definition at line 306 of file KDChartWidget.cpp.

References d.

```
307 {
308     return d->m_chart.legend();
309 }
```

7.69.4.16 LineDiagram ∗ Widget::lineDiagram ()

If the current diagram is a LineDiagram, it is returnd; otherwise 0 is returned.

This function provides type-safe casting.

Definition at line 363 of file KDChartWidget.cpp.

References diagram().

```
{
    return dynamic_cast<LineDiagram*>( diagram() );
}
```

7.69.4.17 PieDiagram ∗ Widget::pieDiagram ()

If the current diagram is a PieDiagram, it is returnd; otherwise 0 is returned.

This function provides type-safe casting.

Definition at line 367 of file KDChartWidget.cpp.

References diagram().

```
{
    return dynamic_cast<PieDiagram*>( diagram() );
}
```

7.69.4.18 PolarDiagram ∗ Widget::polarDiagram ()

If the current diagram is a PolarDiagram, it is returnd; otherwise 0 is returned.

This function provides type-safe casting.

Definition at line 375 of file KDChartWidget.cpp.

References diagram().

```
{
    return dynamic_cast<PolarDiagram*>( diagram() );
}
```

7.69.4.19 void Widget::replaceHeaderFooter (HeaderFooter ∗ *header*, HeaderFooter ∗ *oldHeader* = 0)

Replaces the old header (or footer, resp.), or appends the new header or footer, it there is none yet.

Parameters:

headerFooter The header or footer to be used instead of the old one. This parameter must not be zero, or the method will do nothing.

oldHeaderFooter The header or footer to be removed by the new one. This header or footer will be deleted automatically. If the parameter is omitted, the very first header or footer will be replaced. In case, there was no header and no footer yet, the new header or footer will just be added.

Note:

If you want to re-use the old header or footer, call takeHeaderFooter and addHeaderFooter, instead of using replaceHeaderFooter.

See also:

addHeaderFooter, takeHeaderFooter

Definition at line 292 of file KDChartWidget.cpp.

References d, and KDChart::HeaderFooter::setParent().

```
{
    header->setParent( &d->m_chart );
    d->m_chart.replaceHeaderFooter( header, oldHeader );
}
```

7.69.4.20 void Widget::replaceLegend (Legend ∗ *legend*, Legend ∗ *oldLegend* = 0)

Definition at line 339 of file KDChartWidget.cpp.

References d, diagram(), and KDChart::Legend::setDiagram().

```
{
    legend->setDiagram( diagram() );
    legend->setParent( &d->m_chart );
    d->m_chart.replaceLegend( legend, oldLegend );
}
```

7.69.4.21 void Widget::resetData ()

Resets all data.

Definition at line 175 of file KDChartWidget.cpp.

References d.

```
{
    d->m_model.clear();
    d->usedDatasetWidth = 0;
}
```

7.69.4.22 RingDiagram ∗ Widget::ringDiagram ()

If the current diagram is a RingDiagram, it is returnd; otherwise 0 is returned.

This function provides type-safe casting.

Definition at line 371 of file KDChartWidget.cpp.

References diagram().

```
{
    return dynamic_cast<RingDiagram*>( diagram() );
}
```

7.69.4.23 void Widget::setDataCell (int *row*, int *column*, QPair< double, double > *data*)

Sets the data for a given column using an (X, Y) QPair of doubles.

Definition at line 156 of file KDChartWidget.cpp.

References d.

```
{
    if ( ! checkDatasetWidth( 2 ))
        return;
160
161     QStandardItemModel & model = d->m_model;
162
163     justifyModelSize( row + 1, (column + 1) * 2 );
164
165     QModelIndex index = model.index( row, column * 2 );
166     model.setData( index, QVariant( data.first ), Qt::DisplayRole );
167
168     index = model.index( row, column * 2 + 1 );
169     model.setData( index, QVariant( data.second ), Qt::DisplayRole );
170 }
```

7.69.4.24 void Widget::setDataCell (int *row*, int *column*, double *data*)

Sets the Y value data for a given cell.

Definition at line 143 of file KDChartWidget.cpp.

References d.

```
144 {
145     if ( ! checkDatasetWidth( 1 ) )
146         return;
147
148     QStandardItemModel & model = d->m_model;
149
150     justifyModelSize( row + 1, column + 1 );
151
152     const QModelIndex index = model.index( row, column );
153     model.setData( index, QVariant( data ), Qt::DisplayRole );
154 }
```

7.69.4.25 void Widget::setDataset (int *column*, const QVector< QPair< double, double > > & *data*, const QString & *title* = QString())

Sets the data in the given column using a QVector of QPairs of double for the (X, Y) values.

Definition at line 120 of file KDChartWidget.cpp.

References d.

```
121 {
122     if ( ! checkDatasetWidth( 2 ))
123         return;
124
125     QStandardItemModel & model = d->m_model;
126
127     justifyModelSize( data.size(), (column + 1) * 2 );
128
129     for( int i = 0; i < data.size(); ++i )
130     {
131         QModelIndex index = model.index( i, column * 2 );
132         model.setData( index, QVariant( data[i].first ), Qt::DisplayRole );
133
134         index = model.index( i, column * 2 + 1 );
135         model.setData( index, QVariant( data[i].second ), Qt::DisplayRole );
136     }
137     if ( ! title.isEmpty() ){
138         model.setHeaderData( column * 2,   Qt::Horizontal, QVariant( title ) );
139         model.setHeaderData( column * 2+1, Qt::Horizontal, QVariant( title ) );
140     }
141 }
```

7.69.4.26 void Widget::setDataset (int *column*, const QVector< double > & *data*, const QString & *title* = QString())

Sets the data in the given column using a QVector of double for the Y values.

Definition at line 102 of file KDChartWidget.cpp.

References d.

```
{
    if ( ! checkDatasetWidth( 1 ) )
        return;

    QStandardItemModel & model = d->m_model;

    justifyModelSize( data.size(), column + 1 );

    for( int i = 0; i < data.size(); ++i )
    {
        const QModelIndex index = model.index( i, column );
        model.setData( index, QVariant( data[i] ), Qt::DisplayRole );
    }
    if ( ! title.isEmpty() )
        model.setHeaderData( column, Qt::Horizontal, QVariant( title ) );
}
```

7.69.4.27 void Widget::setGlobalLeadingBottom (int *leading*)

Sets the bottom leading (border).

Definition at line 240 of file KDChartWidget.cpp.

References d.

```
{
    d->m_chart.setGlobalLeadingBottom( leading );
}
```

7.69.4.28 void Widget::setGlobalLeadingLeft (int *leading*)

Sets the left leading (border).

Definition at line 192 of file KDChartWidget.cpp.

References d.

```
{
    d->m_chart.setGlobalLeadingLeft( leading );
}
```

7.69.4.29 void Widget::setGlobalLeadingRight (int *leading*)

Sets the right leading (border).

Definition at line 224 of file KDChartWidget.cpp.

References d.

```
{
    d->m_chart.setGlobalLeadingRight( leading );
}
```

7.69.4.30 void Widget::setGlobalLeadingTop (int *leading*)

Sets the top leading (border).

Definition at line 208 of file KDChartWidget.cpp.

References d.

```
{
    d->m_chart.setGlobalLeadingTop( leading );
}
```

7.69.4.31 void Widget::setSubType (SubType *subType*)

Sets the type of the chart without changing the main type.

Make sure to use a sub-type that matches the main type, so e.g. setting sub-type Rows makes sense for Bar charts only, and it will be ignored for all other chart types.

See also:

KDChartBarDiagram::BarType, KDChartLineDiagram::LineType
KDChartPieDiagram::PieType, KDChartRingDiagram::RingType
KDChartPolarDiagram::PolarType

Definition at line 462 of file KDChartWidget.cpp.

References diagram(), and SET_SUB_TYPE.

```
{
    BarDiagram*  barDia  = qobject_cast< BarDiagram* >(   diagram() );
    LineDiagram* lineDia = qobject_cast< LineDiagram* >(  diagram() );

//FIXME(khz): Add the impl for these chart types - or remove them from here:
//    PieDiagram*   pieDia   = qobject_cast< PieDiagram* >(   diagram() );
//    PolarDiagram* polarDia = qobject_cast< PolarDiagram* >( diagram() );
//    RingDiagram*  ringDia  = qobject_cast< RingDiagram* >(  diagram() );

#define SET_SUB_TYPE(DIAGRAM, SUBTYPE) \
{ \
    if( DIAGRAM ) \
        DIAGRAM->setType( SUBTYPE ); \
}
    switch ( subType )
    {
        case Normal:
           SET_SUB_TYPE( barDia,  BarDiagram::Normal );
           SET_SUB_TYPE( lineDia, LineDiagram::Normal );
           break;
        case Stacked:
           SET_SUB_TYPE( barDia,  BarDiagram::Stacked );
           SET_SUB_TYPE( lineDia, LineDiagram::Stacked );
           break;
        case Percent:
           SET_SUB_TYPE( barDia,  BarDiagram::Percent );
           SET_SUB_TYPE( lineDia, LineDiagram::Percent );
           break;
        case Rows:
           SET_SUB_TYPE( barDia, BarDiagram::Rows );
           break;
        default:
           Q_ASSERT_X ( false,
                        "Widget::setSubType", "Sub-type not supported!" );
           break;
    }
//    coordinatePlane()->show();
}
```

7.69.4.32 SubType KDChart::Widget::subType () const

Returns the sub-type of the chart.

7.69.4.33 void Widget::takeHeaderFooter (HeaderFooter ∗ *header*)

Remove the header (or footer, resp.) from the widget, without deleting it.

The chart no longer owns the header or footer, so it is the caller's responsibility to delete the header or footer.

See also:
addHeaderFooter, replaceHeaderFooter

Definition at line 298 of file KDChartWidget.cpp.

References d.

```
{
    d->m_chart.takeHeaderFooter( header );
}
```

7.69.4.34 void Widget::takeLegend (Legend ∗ *legend*)

Definition at line 346 of file KDChartWidget.cpp.

References d.

```
{
    d->m_chart.takeLegend( legend );
}
```

7.69.4.35 Widget::ChartType Widget::type () const

Returns the type of the chart.

Definition at line 505 of file KDChartWidget.cpp.

References Bar, Line, NoType, Pie, Polar, and Ring.

```
{
    // PENDING(christoph) save the type out-of-band:
    AbstractDiagram * const dia = const_cast<Widget*>( this )->diagram();
    if ( qobject_cast< BarDiagram* >( dia ) )
        return Bar;
    else if ( qobject_cast< LineDiagram* >( dia ) )
        return Line;
    else if( qobject_cast< PieDiagram* >( dia ) )
        return Pie;
    else if( qobject_cast< PolarDiagram* >( dia ) )
        return Polar;
    else if( qobject_cast< RingDiagram* >( dia ) )
        return Ring;
    else
        return NoType;
}
```

7.69.5 Member Data Documentation

7.69.5.1 public int int int KDChart::Widget::bottom

Definition at line 96 of file KDChartWidget.h.

7.69.5.2 public KDChart::Widget::Q_SLOTS

Definition at line 216 of file KDChartWidget.h.

7.69.5.3 public int int KDChart::Widget::right

Definition at line 96 of file KDChartWidget.h.

7.69.5.4 Widget::SubType Widget::subType = Normal)

Definition at line 523 of file KDChartWidget.cpp.

```
{
    // PENDING(christoph) save the type out-of-band:
    Widget::SubType retVal = Normal;

    AbstractDiagram * const dia = const_cast<Widget*>( this )->diagram();
    BarDiagram*  barDia  = qobject_cast< BarDiagram* >(   dia );
    LineDiagram* lineDia = qobject_cast< LineDiagram* >(  dia );

//FIXME(khz): Add the impl for these chart types - or remove them from here:
//    PieDiagram*   pieDia   = qobject_cast< PieDiagram* >(   diagram() );
//    PolarDiagram* polarDia = qobject_cast< PolarDiagram* >( diagram() );
//    RingDiagram*  ringDia  = qobject_cast< RingDiagram* >(  diagram() );

#define TEST_SUB_TYPE(DIAGRAM, INTERNALSUBTYPE, SUBTYPE) \
{ \
    if( DIAGRAM && DIAGRAM->type() == INTERNALSUBTYPE ) \
        retVal = SUBTYPE; \
}
    const Widget::ChartType mainType = type();
    switch ( mainType )
    {
        case Bar:
           TEST_SUB_TYPE( barDia, BarDiagram::Normal,  Normal );
           TEST_SUB_TYPE( barDia, BarDiagram::Stacked, Stacked );
           TEST_SUB_TYPE( barDia, BarDiagram::Percent, Percent );
           TEST_SUB_TYPE( barDia, BarDiagram::Rows,    Rows );
           break;
        case Line:
           TEST_SUB_TYPE( lineDia, LineDiagram::Normal,  Normal );
           TEST_SUB_TYPE( lineDia, LineDiagram::Stacked, Stacked );
           TEST_SUB_TYPE( lineDia, LineDiagram::Percent, Percent );
           break;
        case Pie:
           // no impl. yet
           break;
        case Polar:
           // no impl. yet
           break;
        case Ring:
           // no impl. yet
           break;
        default:
           Q_ASSERT_X ( false,
                        "Widget::subType", "Chart type not supported!" );
           break;
    }
    return retVal;
}
```

7.69.5.5 public int KDChart::Widget::top

Definition at line 96 of file KDChartWidget.h.

The documentation for this class was generated from the following files:

- KDChartWidget.h
- KDChartWidget.cpp

7.70 KDChart::ZoomParameters Class Reference

`#include <KDChartZoomParameters.h>`

Collaboration diagram for KDChart::ZoomParameters:

Public Member Functions

- const QPointF center () const
- void setCenter (QPointF center)
- ZoomParameters ()

Public Attributes

- double xCenter
- double xFactor
- double yCenter
- double yFactor

7.70.1 Constructor & Destructor Documentation

7.70.1.1 KDChart::ZoomParameters::ZoomParameters ()

Definition at line 7 of file KDChartZoomParameters.h.

References xCenter, xFactor, yCenter, and yFactor.

```
    : xFactor( 1.0 ),
      yFactor( 1.0 ),
       xCenter( 0.5 ),
       yCenter( 0.5)
     {
     }
```

7.70.2 Member Function Documentation

7.70.2.1 const QPointF KDChart::ZoomParameters::center () const

Definition at line 20 of file KDChartZoomParameters.h.

References xCenter, and yCenter.

```
     {
         return QPointF( xCenter, yCenter );
     }
```

7.70.2.2 void KDChart::ZoomParameters::setCenter (QPointF *center*)

Definition at line 15 of file KDChartZoomParameters.h.

References xCenter, and yCenter.

```
        {
            xCenter = center.x();
            yCenter = center.y();
        }
```

7.70.3 Member Data Documentation

7.70.3.1 double KDChart::ZoomParameters::xCenter

Definition at line 28 of file KDChartZoomParameters.h.

Referenced by center(), setCenter(), and ZoomParameters().

7.70.3.2 double KDChart::ZoomParameters::xFactor

Definition at line 25 of file KDChartZoomParameters.h.

Referenced by ZoomParameters().

7.70.3.3 double KDChart::ZoomParameters::yCenter

Definition at line 29 of file KDChartZoomParameters.h.

Referenced by center(), setCenter(), and ZoomParameters().

7.70.3.4 double KDChart::ZoomParameters::yFactor

Definition at line 26 of file KDChartZoomParameters.h.

Referenced by ZoomParameters().

The documentation for this class was generated from the following file:

- KDChartZoomParameters.h

Chapter 8

KD Chart 2 File Documentation

8.1 KDChartAbstractArea.cpp File Reference

`#include "KDChartAbstractArea.h"`

`#include "KDChartAbstractArea_p.h"`

`#include <qglobal.h>`

`#include <QPainter>`

`#include <QRect>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartAbstractArea.cpp:

Defines

- #define d (d_func())

8.1.1 Define Documentation

8.1.1.1 #define d (d_func())

Definition at line 39 of file KDChartAbstractArea.cpp.

Referenced by KDChart::AbstractCoordinatePlane::AbstractCoordinatePlane(), KDChart::AbstractThreeDAttributes::AbstractThreeDAttributes(), KDChart::Legend::activateTheLayout(), KDChart::AbstractCartesianDiagram::addAxis(), KDChart::Chart::addCoordinatePlane(), KDChart::Legend::addDiagram(), KDChart::AbstractCoordinatePlane::addDiagram(), KDChart::Widget::addHeaderFooter(), KDChart::Chart::addHeaderFooter(), KDChart::Widget::addLegend(), KDChart::Chart::addLegend(), KDChart::CartesianCoordinatePlane::adjustHorizontalRangeToData(), KDChart::CartesianCoordinatePlane::adjustVerticalRangeToData(), KDChart::Legend::alignment(), KDChart::Widget::allHeadersFooters(), KDChart::Widget::allLegends(), KDChart::AbstractDiagram::allowOverlappingDataValueTexts(), KDChart::ThreeDBarAttributes::angle(), KDChart::PolarCoordinatePlane::angleUnit(), KDChart::AbstractDiagram::antiAliasing(), KDChart::AbstractDiagram::attributesModel(), KDChart::AbstractDiagram::attributesModelRootIndex(), KDChart::CartesianCoordinatePlane::autoAdjustGridToZoom(), KDChart::CartesianCoordinatePlane::autoAdjustHorizontalRangeToData(), KDChart::CartesianCoordinatePlane::autoAdjustVerticalRangeToData(), KDChart::AbstractCartesianDiagram::axes(), KDChart::CartesianCoordinatePlane::axesCalcModeX(), KDChart::CartesianCoordinatePlane::axesCalcModeY(), KDChart::Chart::backgroundAttributes(), KDChart::AbstractAreaBase::backgroundAttributes(), KDChart::BarDiagram::barAttributes(), KDChart::AbstractArea::bottomOverlap(), KDChart::Legend::brush(), KDChart::Legend::brushes(), KDChart::Legend::buildLegend(), KDChart::LineDiagram::calculateDataBoundaries(), KDChart::BarDiagram::calculateDataBoundaries(), KDChart::CartesianCoordinatePlane::calculateRawDataBoundingRect(), KDChart::RingDiagram::clone(), KDChart::PolarDiagram::clone(), KDChart::PieDiagram::clone(), KDChart::LineDiagram::clone(), KDChart::Legend::clone(), KDChart::HeaderFooter::clone(), KDChart::BarDiagram::clone(), KDChart::PolarDiagram::closeDatasets(), KDChart::AbstractAxis::connectSignals(), KDChart::Legend::constDiagrams(), KDChart::Widget::coordinatePlane(), KDChart::Chart::coordinatePlane(), KDChart::AbstractDiagram::coordinatePlane(), KDChart::AbstractAxis::coordinatePlane(), KDChart::Chart::coordinatePlaneLayout(), KDChart::Chart::coordinatePlanes(), KDChart::AbstractAxis::createObserver(), KDChart::AbstractDiagram::dataBoundaries(), KDChart::AbstractDiagram::dataChanged(), KDChart::Legend::datasetCount(), KDChart::AbstractDiagram::datasetDimension(), KDChart::AbstractAxis::deleteObserver(), KDChart::AbstractThreeDAttributes::depth(), KDChart::Legend::diagram(), KDChart::AbstractCoordinatePlane::diagram(), KDChart::AbstractAxis::diagram(), KDChart::Legend::diagrams(), KDChart::AbstractCoordinatePlane::diagrams(), KDChart::CartesianCoordinatePlane::doesIsometricScaling(), KDChart::AbstractDiagram::doItemsLayout(), KDChart::CartesianCoordinatePlane::doneSetZoomCenter(), KDChart::CartesianCoordinatePlane::doneSetZoomFactorX(), KDChart::CartesianCoordinatePlane::doneSetZoomFactorY(), KDChart::PieAttributes::explode(), KDChart::PieAttributes::explodeFactor(), KDChart::Widget::firstHeaderFooter(), KDChart::Legend::floatingPosition(), KDChart::Chart::frameAttributes(), KDChart::AbstractAreaBase::frameAttributes(), KDChart::CartesianAxis::geometry(), KDChart::AbstractCoordinatePlane::geometry(), KDChart::AbstractAreaBase::getFrameLeadings(), KDChart::AbstractCoordinatePlane::globalGridAttributes(), KDChart::Widget::globalLeadingBottom(), KDChart::Chart::globalLeadingBottom(), KDChart::Widget::globalLeadingLeft(), KDChart::Chart::globalLeadingLeft(), KDChart::Widget::globalLeadingRight(), KDChart::Chart::globalLeadingRight(), KDChart::Widget::globalLeadingTop(), KDChart::Chart::globalLeadingTop(), KDChart::AbstractPieDiagram::granularity(), KDChart::PolarCoordinatePlane::gridAttributes(), KDChart::CartesianCoordinatePlane::gridAttributes(), KDChart::AbstractCoordinatePlane::gridDimensionsList(), KDChart::CartesianAxis::hasDefaultTitleTextAttributes(), KDChart::PolarCoordinatePlane::hasOwnGridAttributes(), KDChart::CartesianCoordinatePlane::hasOwnGridAttributes(), KDChart::Chart::headerFooter(), KDChart::Chart::headerFooters(), KDChart::CartesianCoordinatePlane::horizontalRange(), KDChart::AbstractThreeDAttributes::isEnabled(), KDChart::AbstractCoordinatePlane::isVisiblePoint(), KDChart::AbstractAxis::labels(), KDChart::PolarCoordinatePlane::layoutDiagrams(), KDChart::CartesianCoordinatePlane::layoutDiagrams(), KDChart::CartesianAxis::layoutPlanes(), KDChart::AbstractArea::leftOverlap(), KDChart::Widget::legend(), KDChart::Legend::Legend(), KDChart::Chart::legend(), KDChart::Chart::legends(), KDChart::Legend::legendStyle(), KDChart::LineDiagram::lineAttributes(), KDChart::ThreeDLineAttributes::lineXRotation(), KDChart::ThreeDLineAttributes::lineYRotation(), KDChart::Legend::markerAttributes(), KDChart::CartesianAxis::maximumSize(), KDChart::Chart::mousePressEvent(), KDChart::AbstractCoordinatePlane::mousePressEvent(), KDChart::LineDiagram::numberOfAbscissaSegments(), KDChart::BarDiagram::numberOfAbscissaSegments(), KDChart::LineDiagram::numberOfOrdinateSegments(), KDChart::BarDiagram::numberOfOrdinateSegments(), KDChart::AbstractAxis::observedBy(), KDChart::ThreeDPieAttributes::operator=(), KDChart::ThreeDLineAttributes::operator=(), KDChart::ThreeDBarAttributes::operator=(), KDChart::PieAttributes::operator=(), KDChart::AbstractThreeDAttributes::operator=(), KDChart::Legend::orientation(), KDChart::PolarCoordinatePlane::paint(), KDChart::PieDiagram::paint(), KDChart::LineDiagram::paint(), KDChart::Legend::paint(), KDChart::Chart::paint(), KDChart::CartesianCoordinatePlane::paint(), KDChart::CartesianAxis::paint(), KDChart::BarDiagram::paint(), KDChart::AbstractArea::paintAll(), KDChart::AbstractAreaBase::paintBackground(), KDChart::CartesianAxis::paintCtx(), KDChart::AbstractDiagram::paintDataValueText(), KDChart::Chart::paintEvent(), KDChart::AbstractAreaWidget::paintEvent(), KDChart::Abstract-

AreaBase::paintFrame(), KDChart::AbstractAreaWidget::paintIntoRect(), KDChart::AbstractCoordinatePlane::parent(), KDChart::Legend::pen(), KDChart::Legend::pens(), KDChart::AbstractDiagram::percentMode(), KDChart::PieAttributes::PieAttributes(), KDChart::AbstractPieDiagram::pieAttributes(), KDChart::Legend::position(), KDChart::HeaderFooter::position(), KDChart::CartesianAxis::position(), KDChart::PolarCoordinatePlane::radiusUnit(), KDChart::Legend::referenceArea(), KDChart::AbstractCoordinatePlane::referenceCoordinatePlane(), KDChart::AbstractCartesianDiagram::referenceDiagram(), KDChart::AbstractCartesianDiagram::referenceDiagramOffset(), KDChart::RingDiagram::relativeThickness(), KDChart::Chart::reLayoutFloatingLegends(), KDChart::Legend::removeDiagram(), KDChart::Legend::removeDiagrams(), KDChart::Chart::replaceCoordinatePlane(), KDChart::Legend::replaceDiagram(), KDChart::AbstractCoordinatePlane::replaceDiagram(), KDChart::Widget::replaceHeaderFooter(), KDChart::Chart::replaceHeaderFooter(), KDChart::Widget::replaceLegend(), KDChart::Chart::replaceLegend(), KDChart::Widget::resetData(), KDChart::LineDiagram::resetLineAttributes(), KDChart::Legend::resetTexts(), KDChart::CartesianAxis::resetTitleTextAttributes(), KDChart::PolarCoordinatePlane::resizeEvent(), KDChart::Chart::resizeEvent(), KDChart::Legend::resizeLayout(), KDChart::AbstractArea::rightOverlap(), KDChart::PolarDiagram::rotateCircularLabels(), KDChart::Legend::setAlignment(), KDChart::AbstractDiagram::setAllowOverlappingDataValueTexts(), KDChart::ThreeDBarAttributes::setAngle(), KDChart::AbstractDiagram::setAntiAliasing(), KDChart::AbstractDiagram::setAttributesModel(), KDChart::AbstractDiagram::setAttributesModelRootIndex(), KDChart::CartesianCoordinatePlane::setAutoAdjustGridToZoom(), KDChart::CartesianCoordinatePlane::setAutoAdjustHorizontalRangeToData(), KDChart::CartesianCoordinatePlane::setAutoAdjustVerticalRangeToData(), KDChart::CartesianCoordinatePlane::setAxesCalcModes(), KDChart::CartesianCoordinatePlane::setAxesCalcModeX(), KDChart::CartesianCoordinatePlane::setAxesCalcModeY(), KDChart::Chart::setBackgroundAttributes(), KDChart::AbstractAreaBase::setBackgroundAttributes(), KDChart::BarDiagram::setBarAttributes(), KDChart::Legend::setBrush(), KDChart::Legend::setBrushesFromDiagram(), KDChart::PolarDiagram::setCloseDatasets(), KDChart::Legend::setColor(), KDChart::AbstractDiagram::setCoordinatePlane(), KDChart::AbstractDiagram::setDataBoundariesDirty(), KDChart::Widget::setDataCell(), KDChart::Widget::setDataset(), KDChart::AbstractDiagram::setDatasetDimension(), KDChart::AbstractDiagram::setDataValueAttributes(), KDChart::AbstractThreeDAttributes::setDepth(), KDChart::AbstractThreeDAttributes::setEnabled(), KDChart::PieAttributes::setExplode(), KDChart::PieAttributes::setExplodeFactor(), KDChart::Legend::setFloatingPosition(), KDChart::Chart::setFrameAttributes(), KDChart::AbstractAreaBase::setFrameAttributes(), KDChart::CartesianAxis::setGeometry(), KDChart::AbstractCoordinatePlane::setGeometry(), KDChart::AbstractCoordinatePlane::setGlobalGridAttributes(), KDChart::Chart::setGlobalLeading(), KDChart::Widget::setGlobalLeadingBottom(), KDChart::Chart::setGlobalLeadingBottom(), KDChart::Widget::setGlobalLeadingLeft(), KDChart::Chart::setGlobalLeadingLeft(), KDChart::Widget::setGlobalLeadingRight(), KDChart::Chart::setGlobalLeadingRight(), KDChart::Widget::setGlobalLeadingTop(), KDChart::Chart::setGlobalLeadingTop(), KDChart::AbstractPieDiagram::setGranularity(), KDChart::PolarCoordinatePlane::setGridAttributes(), KDChart::CartesianCoordinatePlane::setGridAttributes(), KDChart::AbstractCoordinatePlane::setGridNeedsRecalculate(), KDChart::AbstractDiagram::setHidden(), KDChart::CartesianCoordinatePlane::setHorizontalRange(), KDChart::CartesianCoordinatePlane::setIsometricScaling(), KDChart::AbstractAxis::setLabels(), KDChart::Legend::setLegendStyle(), KDChart::LineDiagram::setLineAttributes(), KDChart::ThreeDLineAttributes::setLineXRotation(), KDChart::ThreeDLineAttributes::setLineYRotation(), KDChart::Legend::setMarkerAttributes(), KDChart::AbstractDiagram::setModel(), KDChart::Legend::setOrientation(), KDChart::AbstractCoordinatePlane::setParent(), KDChart::Legend::setPen(), KDChart::AbstractDiagram::setPercentMode(), KDChart::AbstractPieDiagram::setPieAttributes(), KDChart::Legend::setPosition(), KDChart::HeaderFooter::setPosition(), KDChart::CartesianAxis::setPosition(), KDChart::Legend::setReferenceArea(), KDChart::AbstractCoordinatePlane::setReferenceCoordinatePlane(), KDChart::AbstractCartesianDiagram::setReferenceDiagram(), KDChart::RingDiagram::setRelativeThickness(), KDChart::AbstractDiagram::setRootIndex(), KDChart::PolarDiagram::setRotateCircularLabels(), KDChart::AbstractAxis::setShortLabels(), KDChart::PolarDiagram::setShowDelimitersAtPosition(), KDChart::PolarDiagram::setShowLabelsAtPosition(), KDChart::Legend::setShowLines(), KDChart::Legend::setSpacing(), KDChart::PolarCoordinatePlane::setStartPosition(), KDChart::Legend::setText(), KDChart::Legend::setTextAttributes(), KDChart::AbstractAxis::setTextAttributes(), KDChart::BarDiagram::setThreeDBarAttributes(), KDChart::LineDiagram::setThreeDLineAttributes(), KDChart::AbstractPieDiagram::setThreeDPieAttributes(), KDChart::Legend::setTitleText(), KDChart::CartesianAxis::setTitleText(), KDChart::Legend::setTitleTextAttributes(), KDChart::CartesianAxis::setTitleTextAttributes(), KDChart::LineDiagram::setType(), KDChart::HeaderFooter::setType(), KDChart::BarDiagram::setType(), KDChart::Legend::setUseAutomaticMarkerSize(), KDChart::ThreeDPieAttributes::setUseShadowColors(), KDChart::ThreeDBarAttributes::setUseShadowColors(), KDChart::CartesianCoordinatePlane::setVerticalRange(), KDChart::PolarCoordinatePlane::setZoomCenter(), KDChart::PolarCoordinatePlane::setZoomFactorX(), KDChart::PolarCoordinatePlane::setZoomFactorY(), KDChart::AbstractAxis::shortLabels(), KDChart::PolarDiagram::showDelimitersAtPosition(), KDChart::PolarDiagram::showLabelsAtPosition(), KDChart::Legend::showLines(), KDChart::Legend::sizeHint(), KDChart::Legend::spacing(), KDChart::PolarCoordinatePlane::startPosition(), KDChart::AbstractCartesianDiagram::takeAxis(), KDChart::Chart::takeCoordinatePlane(), KDChart::AbstractCoordinatePlane::takeDiagram(), KDChart::Widget::takeHeaderFooter(), KDChart::Chart::takeHeaderFooter(), KDChart::Widget::takeLegend(), KDChart::Chart::takeLegend(), KDChart::Legend::text(), KDChart::Legend::textAttributes(), KDChart::AbstractAxis::textAttributes(), KDChart::Legend::texts(), KDChart::ThreeDBarAttributes::ThreeDBarAttributes(), KDChart::BarDiagram::threeDBarAttributes(), KDChart::LineDiagram::threeDItemDepth(), KDChart::BarDiagram::threeDItemDepth(), KDChart::ThreeDLineAttributes::ThreeDLineAttributes(), KDChart::LineDiagram::threeDLineAttributes(), KDChart::ThreeDPieAttributes::ThreeDPieAttributes(), KDChart::AbstractPieDiagram::threeDPieAttributes(), KDChart::Legend::titleText(), KDChart::CartesianAxis::titleText(), KDChart::Legend::titleTextAttributes(), KDChart::CartesianAxis::titleTextAttributes(), KDChart::AbstractArea::topOverlap(), KDChart::PolarCoordinatePlane::translate(), KDChart::CartesianCoordinatePlane::translate(), KDChart::CartesianCoordinatePlane::translateBack(), KDChart::PolarCoordinatePlane::translatePolar(), KDChart::LineDiagram::type(), KDChart::HeaderFooter::type(), KDChart::BarDiagram::type(), KDChart::AbstractDiagram::update(), KDChart::Legend::useAutomaticMarkerSize(), KDChart::AbstractDiagram::useDefaultColors(), KDChart::AbstractDiagram::useRainbowColors(), KDChart::AbstractDiagram::usesExternalAttributesModel(), KDChart::ThreeDPieAttributes::useShadowColors(), KDChart::ThreeDBarAttributes::useShadowColors(), KDChart::AbstractDiagram::useSubduedColors(), KDChart::AbstractThreeDAttributes::validDepth(), KDChart::AbstractDiagram::valueForCell(), KDChart::LineDiagram::valueForCellTesting(), KDChart::CartesianCoordinatePlane::verticalRange(), KDChart::PolarCoordinatePlane::zoomCenter(), KDChart::CartesianCoordinatePlane::zoomCenter(), KDChart::PolarCoordinatePlane::zoomFactorX(), KDChart::CartesianCoordinatePlane::zoomFactorX(), KDChart::PolarCoordinatePlane::zoomFactorY(), KDChart::CartesianCoordinatePlane::zoomFactorY(), KDChart::AbstractAxis::∼AbstractAxis(), KDChart::AbstractCartesianDiagram::∼AbstractCartesianDiagram(), and KDChart::CartesianAxis::∼CartesianAxis().

8.2 KDChartAbstractArea.h File Reference

`#include <QObject>`

`#include "KDChartGlobal.h"`

`#include "KDChartAbstractAreaBase.h"`

`#include "KDChartLayoutItems.h"`

Include dependency graph for KDChartAbstractArea.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.3 KDChartAbstractAreaBase.cpp File Reference

```
#include "KDChartAbstractAreaBase.h"
#include "KDChartAbstractAreaBase_p.h"
#include <KDChartBackgroundAttributes.h>
#include <KDChartFrameAttributes.h>
#include <KDChartTextAttributes.h>
#include "KDChartPainterSaver_p.h"
#include <QPainter>
#include <KDABLibFakes>
```

Include dependency graph for KDChartAbstractAreaBase.cpp:

Defines

- #define d d_func()

8.3.1 Define Documentation

8.3.1.1 #define d d_func()

Definition at line 73 of file KDChartAbstractAreaBase.cpp.

8.4 KDChartAbstractAreaBase.h File Reference

```
#include <QPointF>
#include <QSizeF>
#include <QRectF>
#include "KDChartGlobal.h"
#include "KDChartLayoutItems.h"
#include "KDChartRelativePosition.h"
#include "KDChartAbstractAreaBase.h"
```

Include dependency graph for KDChartAbstractAreaBase.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.5 KDChartAbstractAreaWidget.cpp File Reference

`#include "KDChartAbstractAreaWidget.h"`

`#include "KDChartAbstractAreaWidget_p.h"`

`#include <KDABLibFakes>`

Include dependency graph for KDChartAbstractAreaWidget.cpp:

Defines

- #define d d_func()

8.5.1 Define Documentation

8.5.1.1 #define d d_func()

Definition at line 91 of file KDChartAbstractAreaWidget.cpp.

8.6 KDChartAbstractAreaWidget.h File Reference

`#include <QWidget>`

`#include <QPaintEvent>`

`#include <QPainter>`

`#include <QRect>`

`#include "KDChartAbstractAreaBase.h"`

Include dependency graph for KDChartAbstractAreaWidget.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.7 KDChartAbstractAxis.cpp File Reference

```
#include <QDebug>
#include "KDChartAbstractAxis.h"
#include "KDChartAbstractAxis_p.h"
#include "KDChartAbstractDiagram.h"
#include "KDChartAbstractCartesianDiagram.h"
#include "KDChartEnums.h"
#include "KDChartMeasure.h"
#include <KDABLibFakes>
```

Include dependency graph for KDChartAbstractAxis.cpp:

Defines

- #define d d_func()

8.7.1 Define Documentation

8.7.1.1 #define d d_func()

Definition at line 39 of file KDChartAbstractAxis.cpp.

8.8 KDChartAbstractAxis.h File Reference

```
#include "kdchart_export.h"
#include "KDChartGlobal.h"
#include "KDChartAbstractArea.h"
#include "KDChartTextAttributes.h"
```

Include dependency graph for KDChartAbstractAxis.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.9 KDChartAbstractCartesianDiagram.cpp File Reference

```
#include "KDChartAbstractCartesianDiagram.h"
#include "KDChartAbstractCartesianDiagram_p.h"
#include "KDChartPaintContext.h"
#include <QDebug>
#include <QPainter>
#include <KDABLibFakes>
```

Include dependency graph for KDChartAbstractCartesianDiagram.cpp:

Defines

- #define d d_func()

8.9.1 Define Documentation

8.9.1.1 #define d d_func()

Definition at line 74 of file KDChartAbstractCartesianDiagram.cpp.

8.10 KDChartAbstractCartesianDiagram.h File Reference

```
#include "KDChartCartesianCoordinatePlane.h"
#include "KDChartAbstractDiagram.h"
#include "KDChartCartesianAxis.h"
```

Include dependency graph for KDChartAbstractCartesianDiagram.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.11 KDChartAbstractCoordinatePlane.cpp File Reference

```
#include <QGridLayout>
#include "KDChartChart.h"
#include "KDChartAbstractCoordinatePlane.h"
#include "KDChartAbstractCoordinatePlane_p.h"
#include "KDChartGridAttributes.h"
#include <KDABLibFakes>
```

Include dependency graph for KDChartAbstractCoordinatePlane.cpp:

Defines

- #define d d_func()

8.11.1 Define Documentation

8.11.1.1 #define d d_func()

Definition at line 39 of file KDChartAbstractCoordinatePlane.cpp.

8.12 KDChartAbstractCoordinatePlane.h File Reference

```
#include <QObject>
#include <QList>
#include "KDChartAbstractArea.h"
#include "KDChartAbstractDiagram.h"
#include "KDChartEnums.h"
```

Include dependency graph for KDChartAbstractCoordinatePlane.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.13 KDChartAbstractDiagram.cpp File Reference

```
#include <QPainter>
#include <QDebug>
#include <QApplication>
#include <QAbstractProxyModel>
#include <QStandardItemModel>
#include <QSizeF>
#include "KDChartAbstractCoordinatePlane.h"
#include "KDChartChart.h"
#include "KDChartDataValueAttributes.h"
#include "KDChartTextAttributes.h"
#include "KDChartMarkerAttributes.h"
#include "KDChartAbstractDiagram.h"
#include "KDChartAbstractDiagram_p.h"
#include "KDChartAttributesModel.h"
#include "KDChartAbstractThreeDAttributes.h"
#include "KDChartThreeDLineAttributes.h"
#include <KDABLibFakes>
#include "KDChartAbstractDiagram.moc"
```

Include dependency graph for KDChartAbstractDiagram.cpp:

Namespaces

- namespace KDChart

Defines

- #define d d_func()

8.13.1 Define Documentation

8.13.1.1 #define d d_func()

Definition at line 117 of file KDChartAbstractDiagram.cpp.

8.14 KDChartAbstractDiagram.h File Reference

```
#include <QList>
#include <QRectF>
#include <QAbstractItemView>
#include "KDChartGlobal.h"
#include "KDChartMarkerAttributes.h"
```

Include dependency graph for KDChartAbstractDiagram.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.15 KDChartAbstractPieDiagram.cpp File Reference

```
#include <QMap>
#include "KDChartAbstractPieDiagram.h"
#include "KDChartAbstractPieDiagram_p.h"
#include "KDChartAttributesModel.h"
#include "KDChartPieAttributes.h"
#include "KDChartThreeDPieAttributes.h"
#include <KDABLibFakes>
```

Include dependency graph for KDChartAbstractPieDiagram.cpp:

Defines

- #define d d_func()

8.15.1 Define Documentation

8.15.1.1 #define d d_func()

Definition at line 62 of file KDChartAbstractPieDiagram.cpp.

8.16 KDChartAbstractPieDiagram.h File Reference

`#include "KDChartAbstractPolarDiagram.h"`

Include dependency graph for KDChartAbstractPieDiagram.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.17 KDChartAbstractPolarDiagram.cpp File Reference

`#include "KDChartAbstractPolarDiagram.h"`

`#include "KDChartAbstractPolarDiagram_p.h"`

`#include <KDABLibFakes>`

Include dependency graph for KDChartAbstractPolarDiagram.cpp:

Defines

- #define d d_func()

8.17.1 Define Documentation

8.17.1.1 #define d d_func()

Definition at line 46 of file KDChartAbstractPolarDiagram.cpp.

8.18 KDChartAbstractPolarDiagram.h File Reference

`#include "KDChartPolarCoordinatePlane.h"`

`#include "KDChartAbstractDiagram.h"`

Include dependency graph for KDChartAbstractPolarDiagram.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.19 KDChartAbstractProxyModel.cpp File Reference

`#include "KDChartAbstractProxyModel.h"`

`#include <QDebug>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartAbstractProxyModel.cpp:

Namespaces

- namespace KDChart

8.20 KDChartAbstractProxyModel.h File Reference

`#include <QAbstractProxyModel>`

`#include "KDChartGlobal.h"`

Include dependency graph for KDChartAbstractProxyModel.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.21 KDChartAbstractThreeDAttributes.cpp File Reference

`#include "KDChartAbstractThreeDAttributes.h"`

`#include "KDChartAbstractThreeDAttributes_p.h"`

`#include <QDebug>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartAbstractThreeDAttributes.cpp:

Defines

- #define d d_func()

Functions

- QDebug operator<< (QDebug dbg, const KDChart::AbstractThreeDAttributes &a)

8.21.1 Define Documentation

8.21.1.1 #define d d_func()

Definition at line 33 of file KDChartAbstractThreeDAttributes.cpp.

8.21.2 Function Documentation

8.21.2.1 QDebug operator<< (QDebug *dbg*, const KDChart::AbstractThreeDAttributes & *a*)

Definition at line 116 of file KDChartAbstractThreeDAttributes.cpp.

```
{
    dbg << "enabled="<<a.isEnabled()
        << "depth="<<a.depth();
    return dbg;
}
```

8.22 KDChartAbstractThreeDAttributes.h File Reference

`#include <QMetaType>`

`#include "KDChartGlobal.h"`

Include dependency graph for KDChartAbstractThreeDAttributes.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

Functions

- KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::AbstractThreeDAttributes &)

8.22.1 Function Documentation

8.22.1.1 KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::AbstractThreeDAttributes &)

Definition at line 116 of file KDChartAbstractThreeDAttributes.cpp.

References KDChart::AbstractThreeDAttributes::depth(), and KDChart::AbstractThreeDAttributes::isEnabled().

```
{
    dbg << "enabled="<<a.isEnabled()
        << "depth="<<a.depth();
    return dbg;
}
```

8.23 KDChartAttributesModel.cpp File Reference

```
#include <QDebug>
#include <QPen>
#include <QPointer>
#include "KDChartAttributesModel.h"
#include "KDChartPalette.h"
#include "KDChartGlobal.h"
#include <KDChartTextAttributes>
#include <KDChartFrameAttributes>
#include <KDChartBackgroundAttributes>
#include <KDChartDataValueAttributes>
#include <KDChartMarkerAttributes>
#include <KDChartBarAttributes>
#include <KDChartLineAttributes>
#include <KDChartPieAttributes>
#include <KDChartAbstractThreeDAttributes>
#include <KDChartThreeDBarAttributes>
#include <KDChartThreeDLineAttributes>
#include <KDChartThreeDPieAttributes>
#include <KDChartGridAttributes>
#include <KDABLibFakes>
```

Include dependency graph for KDChartAttributesModel.cpp:

8.24 KDChartAttributesModel.h File Reference

`#include "KDChartAbstractProxyModel.h"`

`#include <QMap>`

`#include <QVariant>`

`#include "KDChartGlobal.h"`

Include dependency graph for KDChartAttributesModel.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.25 KDChartBackgroundAttributes.cpp File Reference

`#include "KDChartBackgroundAttributes.h"`

`#include <QPixmap>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartBackgroundAttributes.cpp:

Defines

- #define d d_func()

Functions

- QDebug operator<< (QDebug dbg, const KDChart::BackgroundAttributes &ba)

8.25.1 Define Documentation

8.25.1.1 #define d d_func()

Definition at line 31 of file KDChartBackgroundAttributes.cpp.

8.25.2 Function Documentation

8.25.2.1 QDebug operator<< (QDebug *dbg*, const KDChart::BackgroundAttributes & *ba*)

Definition at line 150 of file KDChartBackgroundAttributes.cpp.

```
{
    dbg << "KDChart::BackgroundAttributes("
        << "visible="<<ba.isVisible()
        << "brush="<<ba.brush()
        << "pixmapmode="<<ba.pixmapMode()
        << "pixmap="<<ba.pixmap()
        << ")";
    return dbg;
}
```

8.26 KDChartBackgroundAttributes.h File Reference

`#include <QDebug>`

`#include <QMetaType>`

`#include <QBrush>`

`#include "KDChartGlobal.h"`

Include dependency graph for KDChartBackgroundAttributes.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

Functions

- KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::BackgroundAttributes &)
- Q_DECLARE_TYPEINFO (KDChart::BackgroundAttributes, Q_MOVABLE_TYPE)

8.26.1 Function Documentation

8.26.1.1 KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::BackgroundAttributes &)

Definition at line 150 of file KDChartBackgroundAttributes.cpp.

References KDChart::BackgroundAttributes::brush(), KDChart::BackgroundAttributes::isVisible(), KDChart::BackgroundAttributes::pixmap(), and KDChart::BackgroundAttributes::pixmapMode().

```
{
    dbg << "KDChart::BackgroundAttributes("
        << "visible="<<ba.isVisible()
        << "brush="<<ba.brush()
        << "pixmapmode="<<ba.pixmapMode()
        << "pixmap="<<ba.pixmap()
        << ")";
    return dbg;
}
```

8.26.1.2 Q_DECLARE_TYPEINFO (KDChart::BackgroundAttributes, Q_MOVABLE_TYPE)

8.27 KDChartBarAttributes.cpp File Reference

`#include "KDChartBarAttributes.h"`

`#include <qglobal.h>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartBarAttributes.cpp:

Defines

- #define d d_func()

8.27.1 Define Documentation

8.27.1.1 #define d d_func()

Definition at line 31 of file KDChartBarAttributes.cpp.

8.28 KDChartBarAttributes.h File Reference

`#include <QMetaType>`

`#include "KDChartGlobal.h"`

Include dependency graph for KDChartBarAttributes.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.29 KDChartBarDiagram.cpp File Reference

```
#include <QPainter>
#include <QDebug>
#include "KDChartBarDiagram.h"
#include "KDChartBarDiagram_p.h"
#include "KDChartThreeDBarAttributes.h"
#include "KDChartPosition.h"
#include "KDChartAttributesModel.h"
#include "KDChartAbstractGrid.h"
#include <KDABLibFakes>
```

Include dependency graph for KDChartBarDiagram.cpp:

Defines

- #define d d_func()

8.29.1 Define Documentation

8.29.1.1 #define d d_func()

Definition at line 49 of file KDChartBarDiagram.cpp.

8.30 KDChartBarDiagram.h File Reference

`#include "KDChartAbstractCartesianDiagram.h"`

`#include "KDChartBarAttributes.h"`

`#include "KDChartThreeDBarAttributes.h"`

Include dependency graph for KDChartBarDiagram.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.31 KDChartCartesianAxis.cpp File Reference

`#include <cmath>`

`#include <QtDebug>`

`#include <QPainter>`

`#include <QPen>`

`#include <QBrush>`

`#include <QApplication>`

`#include "KDChartPaintContext.h"`

`#include "KDChartChart.h"`

`#include "KDChartCartesianAxis.h"`

`#include "KDChartCartesianAxis_p.h"`

`#include "KDChartAbstractCartesianDiagram.h"`

`#include "KDChartAbstractGrid.h"`

`#include "KDChartPainterSaver_p.h"`

`#include "KDChartLayoutItems.h"`

`#include "KDChartBarDiagram.h"`

`#include <KDABLibFakes>`

Include dependency graph for KDChartCartesianAxis.cpp:

Defines

- #define d (d_func())

Functions

- void calculateNextLabel (qreal &labelValue, qreal step, bool isLogarithmic)
- void calculateOverlap (int i, int first, int last, int measure, bool isBarDiagram, int &firstOverlap, int &lastOverlap)
- bool referenceDiagramIsBarDiagram (const AbstractDiagram ∗diagram)

8.31.1 Define Documentation

8.31.1.1 #define d (d_func())

Definition at line 49 of file KDChartCartesianAxis.cpp.

8.31.2 Function Documentation

8.31.2.1 void calculateNextLabel (qreal & *labelValue*, qreal *step*, bool *isLogarithmic*) `[static]`

Definition at line 347 of file KDChartCartesianAxis.cpp.

Referenced by KDChart::CartesianAxis::paintCtx().

```
348 {
349     if ( isLogarithmic ){
350         labelValue *= 10.0;
351     }else{
352         //qDebug() << "new axis label:" << labelValue << "+" << step << "=" << labelValue+step;
353         labelValue += step;
354     }
355     if( qAbs(labelValue) < 1.0e-15 )
356         labelValue = 0.0;
357 }
```

8.31.2.2 void calculateOverlap (int *i*, int *first*, int *last*, int *measure*, bool *isBarDiagram*, int & *firstOverlap*, int & *lastOverlap*) `[static]`

Definition at line 981 of file KDChartCartesianAxis.cpp.

Referenced by KDChart::CartesianAxis::maximumSize().

```
985 {
986     if( i == first ){
987         if( isBarDiagram ){
988             //TODO(khz): Calculate the amount of left overlap
989             //           for bar diagrams.
990         }else{
991             firstOverlap = measure / 2;
992         }
993     }
994     // we test both bounds in on go: first and last might be equal
995     if( i == last ){
996         if( isBarDiagram ){
997             //TODO(khz): Calculate the amount of right overlap
998             //           for bar diagrams.
999         }else{
1000             lastOverlap = measure / 2;
1001         }
1002     }
1003 }
```

8.31.2.3 bool referenceDiagramIsBarDiagram (const AbstractDiagram ∗ *diagram*) `[static]`

Definition at line 360 of file KDChartCartesianAxis.cpp.

References KDChart::AbstractCartesianDiagram::referenceDiagram().

Referenced by KDChart::CartesianAxis::maximumSize(), and KDChart::CartesianAxis::paintCtx().

```
361 {
362     const AbstractCartesianDiagram * dia =
363             qobject_cast< const AbstractCartesianDiagram * >( diagram );
364     if( dia && dia->referenceDiagram() )
365         dia = dia->referenceDiagram();
366     return qobject_cast< const BarDiagram* >( dia ) != 0;
367 }
```

8.32 KDChartCartesianAxis.h File Reference

`#include <QList>`

`#include "KDChartAbstractAxis.h"`

Include dependency graph for KDChartCartesianAxis.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.33 KDChartCartesianCoordinatePlane.cpp File Reference

```
#include <QFont>
#include <QList>
#include <QtDebug>
#include <QPainter>
#include <QApplication>
#include "KDChartAbstractDiagram.h"
#include "KDChartAbstractCartesianDiagram.h"
#include "KDChartCartesianCoordinatePlane.h"
#include "KDChartCartesianCoordinatePlane_p.h"
#include "CartesianCoordinateTransformation.h"
#include "KDChartGridAttributes.h"
#include "KDChartPaintContext.h"
#include "KDChartPainterSaver_p.h"
#include <KDABLibFakes>
```

Include dependency graph for KDChartCartesianCoordinatePlane.cpp:

Defines

- #define d d_func()

8.33.1 Define Documentation

8.33.1.1 #define d d_func()

Definition at line 46 of file KDChartCartesianCoordinatePlane.cpp.

8.34 KDChartCartesianCoordinatePlane.h File Reference

`#include "KDChartAbstractCoordinatePlane.h"`

Include dependency graph for KDChartCartesianCoordinatePlane.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.35 KDChartChart.cpp File Reference

```
#include <QList>
#include <QtDebug>
#include <QGridLayout>
#include <QLabel>
#include <QHash>
#include <QPainter>
#include <QPaintEvent>
#include <QLayoutItem>
#include <QPushButton>
#include <QApplication>
#include <QEvent>
#include "KDChartChart.h"
#include "KDChartChart_p.h"
#include "KDChartCartesianCoordinatePlane.h"
#include "KDChartAbstractCartesianDiagram.h"
#include "KDChartHeaderFooter.h"
#include "KDChartEnums.h"
#include "KDChartLegend.h"
#include "KDChartLayoutItems.h"
#include <KDChartTextAttributes.h>
#include <KDChartMarkerAttributes>
#include "KDChartPainterSaver_p.h"
#include <KDABLibFakes>
```

Include dependency graph for KDChartChart.cpp:

Defines

- #define ADD_AUTO_SPACER_IF_NEEDED(spacerRow, spacerColumn, hLayoutIsAtTop, hLayout, vLayoutIsAtLeft, vLayout)
- #define ADD_VBOX_WITH_LEGENDS(row, column, align)
- #define d d_func()
- #define SET_ALL_MARGINS_TO_ZERO

Functions

- QHBoxLayout ∗ findOrCreateHBoxLayoutByObjectName (QLayout ∗parentLayout, const char ∗name)
- template<typename T> T ∗ findOrCreateLayoutByObjectName (QLayout ∗parentLayout, const char ∗name)

- QVBoxLayout ∗ findOrCreateVBoxLayoutByObjectName (QLayout ∗parentLayout, const char ∗name)

8.35.1 Define Documentation

8.35.1.1 #define ADD_AUTO_SPACER_IF_NEEDED(spacerRow, spacerColumn, hLayoutIsAtTop, hLayout, vLayoutIsAtLeft, vLayout)

Value:

```
{ \
    if( hLayout || vLayout ) { \
        AutoSpacerLayoutItem * spacer \
                = new AutoSpacerLayoutItem( hLayoutIsAtTop, hLayout, vLayoutIsAtLeft, vLayout ); \
        planeLayout->addItem( spacer, spacerRow, spacerColumn, 1, 1 ); \
        spacer->setParentLayout( planeLayout ); \
        planeLayoutItems << spacer; \
    } \
}
```

8.35.1.2 #define ADD_VBOX_WITH_LEGENDS(row, column, align)

Value:

```
{ \
    QVBoxLayout* innerLayout = new QVBoxLayout(); \
    for (int i = 0; i < count; ++i) { \
        legend = list.at(i); \
        if( legend->alignment() == ( align ) ) \
            innerLayout->addItem( new MyWidgetItem(legend, Qt::AlignLeft) ); \
    } \
    gridLayout->addLayout( innerLayout, row, column, ( align  ) ); \
}
```

8.35.1.3 #define d d_func()

Definition at line 803 of file KDChartChart.cpp.

8.35.1.4 #define SET_ALL_MARGINS_TO_ZERO

Definition at line 57 of file KDChartChart.cpp.

8.35.2 Function Documentation

8.35.2.1 QHBoxLayout∗ findOrCreateHBoxLayoutByObjectName (QLayout ∗ *parentLayout*, const char ∗ *name*) [static]

Definition at line 444 of file KDChartChart.cpp.

```
445 {
446     return findOrCreateLayoutByObjectName<QHBoxLayout>( parentLayout, name );
447 }
```

8.35.2.2 template<typename T> T∗ findOrCreateLayoutByObjectName (QLayout ∗ *parentLayout*, const char ∗ *name*) `[static]`

Definition at line 424 of file KDChartChart.cpp.

```
{
    T *box = qFindChild<T*>( parentLayout, QString::fromLatin1( name ) );
    if ( !box ) {
        box = new T();
        // TESTING(khz): set the margin of all of the layouts to Zero
#if defined SET_ALL_MARGINS_TO_ZERO
        box->setMargin(0);
#endif
        box->setObjectName( QString::fromLatin1( name ) );
        box->setSizeConstraint( QLayout::SetFixedSize );
    }
    return box;
}
```

8.35.2.3 QVBoxLayout∗ findOrCreateVBoxLayoutByObjectName (QLayout ∗ *parentLayout*, const char ∗ *name*) `[static]`

Definition at line 439 of file KDChartChart.cpp.

```
{
    return findOrCreateLayoutByObjectName<QVBoxLayout>( parentLayout, name );
}
```

8.36 KDChartChart.h File Reference

8.36.1 Detailed Description

Declaring the class KDChart::Chart.

Definition in file KDChartChart.h.

`#include <QWidget>`

`#include "kdchart_export.h"`

`#include "KDChartGlobal.h"`

Include dependency graph for KDChartChart.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.37 KDChartDatasetProxyModel.cpp File Reference

```
#include <QtDebug>
#include "KDChartDatasetProxyModel.h"
#include <KDABLibFakes>
```

Include dependency graph for KDChartDatasetProxyModel.cpp:

8.38 KDChartDatasetProxyModel.h File Reference

`#include <QVector>`

`#include <QSortFilterProxyModel>`

`#include "kdchart_export.h"`

Include dependency graph for KDChartDatasetProxyModel.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.39 KDChartDatasetSelector.cpp File Reference

`#include <QtDebug>`

`#include "KDChartDatasetSelector.h"`

`#include "ui_KDChartDatasetSelector.h"`

`#include <KDABLibFakes>`

Include dependency graph for KDChartDatasetSelector.cpp:

8.40 KDChartDatasetSelector.h File Reference

`#include <QFrame>`

`#include "KDChartDatasetProxyModel.h"`

Include dependency graph for KDChartDatasetSelector.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart
- namespace Ui

8.41 KDChartDataValueAttributes.cpp File Reference

`#include <QVariant>`

`#include <QDebug>`

`#include "KDChartDataValueAttributes.h"`

`#include "KDChartRelativePosition.h"`

`#include "KDChartPosition.h"`

`#include <KDChartTextAttributes.h>`

`#include <KDChartFrameAttributes.h>`

`#include <KDChartBackgroundAttributes.h>`

`#include <KDChartMarkerAttributes.h>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartDataValueAttributes.cpp:

Defines

- #define d d_func()
- #define KDCHART_DATA_VALUE_AUTO_DIGITS 4

Functions

- QDebug operator<< (QDebug dbg, const KDChart::DataValueAttributes &val)

8.41.1 Define Documentation

8.41.1.1 #define d d_func()

Definition at line 43 of file KDChartDataValueAttributes.cpp.

8.41.1.2 #define KDCHART_DATA_VALUE_AUTO_DIGITS 4

Definition at line 40 of file KDChartDataValueAttributes.cpp.

8.41.2 Function Documentation

8.41.2.1 QDebug operator<< (QDebug *dbg*, const KDChart::DataValueAttributes & *val*)

Definition at line 324 of file KDChartDataValueAttributes.cpp.

```
{
    dbg << "RelativePosition DataValueAttributes("
        << "visible="<<val.isVisible()
        << "textattributes="<<val.textAttributes()
        << "frameattributes="<<val.frameAttributes()
        << "backgroundattributes="<<val.backgroundAttributes()
        << "decimaldigits="<<val.decimalDigits()
        << "poweroftendivisor="<<val.powerOfTenDivisor()
        << "showinfinite="<<val.showInfinite()
        << "negativerelativeposition="<<val.negativePosition()
        << "positiverelativeposition="<<val.positivePosition()
        << "showRepetitiveDataLabels="<<val.showRepetitiveDataLabels()
        <<")";
    return dbg;
}
```

8.42 KDChartDataValueAttributes.h File Reference

8.42.1 Detailed Description

Declaring the class KDChart::DataValueAttributes.

Definition in file KDChartDataValueAttributes.h.

`#include <Qt>`

`#include <QMetaType>`

`#include "KDChartGlobal.h"`

`#include "KDChartEnums.h"`

`#include "KDChartRelativePosition.h"`

Include dependency graph for KDChartDataValueAttributes.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

Functions

- KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::DataValueAttributes &)
- Q_DECLARE_TYPEINFO (KDChart::DataValueAttributes, Q_MOVABLE_TYPE)

8.42.2 Function Documentation

8.42.2.1 KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::DataValueAttributes &)

Definition at line 324 of file KDChartDataValueAttributes.cpp.

References KDChart::DataValueAttributes::backgroundAttributes(), KDChart::DataValueAttributes::decimalDigits(), KDChart::DataValueAttributes::frameAttributes(), KDChart::DataValueAttributes::isVisible(), KDChart::DataValueAttributes::negativePosition(), KDChart::DataValueAttributes::positivePosition(), KDChart::DataValueAttributes::powerOfTenDivisor(), KDChart::DataValueAttributes::showInfinite(), KDChart::DataValueAttributes::showRepetitiveDataLabels(), and KDChart::DataValueAttributes::textAttributes().

```
325 {
326     dbg << "RelativePosition DataValueAttributes("
327         << "visible="<<val.isVisible()
328         << "textattributes="<<val.textAttributes()
329         << "frameattributes="<<val.frameAttributes()
330         << "backgroundattributes="<<val.backgroundAttributes()
331         << "decimaldigits="<<val.decimalDigits()
332         << "poweroftendivisor="<<val.powerOfTenDivisor()
333         << "showinfinite="<<val.showInfinite()
334         << "negativerelativeposition="<<val.negativePosition()
335         << "positiverelativeposition="<<val.positivePosition()
336         << "showRepetitiveDataLabels="<<val.showRepetitiveDataLabels()
337         <<")";
```

```
    return dbg;
}
```

8.42.2.2 Q_DECLARE_TYPEINFO (KDChart::DataValueAttributes, Q_MOVABLE_TYPE)

8.43 KDChartDiagramObserver.cpp File Reference

`#include <KDChartAbstractDiagram.h>`

`#include <KDChartDiagramObserver.h>`

`#include <KDChartAttributesModel.h>`

`#include <KDABLibFakes>`

`#include <QDebug>`

Include dependency graph for KDChartDiagramObserver.cpp:

8.44 KDChartDiagramObserver.h File Reference

`#include "KDChartGlobal.h"`

`#include <QObject>`

`#include <QPointer>`

`#include <QModelIndex>`

Include dependency graph for KDChartDiagramObserver.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.45 KDChartEnums.h File Reference

8.45.1 Detailed Description

Definition of global enums.

Definition in file KDChartEnums.h.

`#include "KDChartGlobal.h"`

`#include <QRectF>`

`#include <QObject>`

`#include <QVector>`

Include dependency graph for KDChartEnums.h:

This graph shows which files directly or indirectly include this file:

Classes

- class KDChartEnums

  Project global class providing some enums needed both by KDChartParams and by KDChartCustomBox.

8.46 KDChartFrameAttributes.cpp File Reference

`#include "KDChartFrameAttributes.h"`

`#include <KDABLibFakes>`

Include dependency graph for KDChartFrameAttributes.cpp:

Defines

- #define d d_func()

Functions

- QDebug operator<< (QDebug dbg, const KDChart::FrameAttributes &fa)

8.46.1 Define Documentation

8.46.1.1 #define d d_func()

Definition at line 30 of file KDChartFrameAttributes.cpp.

8.46.2 Function Documentation

8.46.2.1 QDebug operator<< (QDebug *dbg*, const KDChart::FrameAttributes & *fa*)

Definition at line 124 of file KDChartFrameAttributes.cpp.

```
{
    dbg << "KDChart::FrameAttributes("
        << "visible="<<fa.isVisible()
        << "pen="<<fa.pen()
        << "padding="<<fa.padding()
        << ")";
    return dbg;
}
```

8.47 KDChartFrameAttributes.h File Reference

`#include <QDebug>`

`#include <QMetaType>`

`#include <QPen>`

`#include "KDChartGlobal.h"`

Include dependency graph for KDChartFrameAttributes.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

Functions

- KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::FrameAttributes &)
- Q_DECLARE_TYPEINFO (KDChart::FrameAttributes, Q_MOVABLE_TYPE)

8.47.1 Function Documentation

8.47.1.1 KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::FrameAttributes &)

Definition at line 124 of file KDChartFrameAttributes.cpp.

References KDChart::FrameAttributes::isVisible(), KDChart::FrameAttributes::padding(), and KDChart::FrameAttributes::pen().

```
{
    dbg << "KDChart::FrameAttributes("
        << "visible="<<fa.isVisible()
        << "pen="<<fa.pen()
        << "padding="<<fa.padding()
        << ")";
    return dbg;
}
```

8.47.1.2 Q_DECLARE_TYPEINFO (KDChart::FrameAttributes, Q_MOVABLE_TYPE)

8.48 KDChartGlobal.h File Reference

`#include <qglobal.h>`

`#include "kdchart_export.h"`

`#include <QtAlgorithms>`

`#include <algorithm>`

`#include <Qt>`

Include dependency graph for KDChartGlobal.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

Defines

- #define KDAB_SET_OBJECT_NAME(x) __kdab__dereference_for_methodcall(x).setObjectName(QLatin1String(#x))
- #define KDCHART_DECLARE_DERIVED_DIAGRAM(X, PLANE)
- #define KDCHART_DECLARE_PRIVATE_BASE_POLYMORPHIC(X)
- #define KDCHART_DECLARE_PRIVATE_BASE_POLYMORPHIC_QWIDGET(X)
- #define KDCHART_DECLARE_PRIVATE_BASE_VALUE(X)
- #define KDCHART_DECLARE_PRIVATE_DERIVED(X)
- #define KDCHART_DECLARE_PRIVATE_DERIVED_PARENT(X, ParentType)
- #define KDCHART_DECLARE_PRIVATE_DERIVED_QWIDGET(X) KDCHART_DECLARE_PRIVATE_DERIVED_PARENT(X, QWidget∗)
- #define KDCHART_DECLARE_SWAP_BASE(X)
- #define KDCHART_DECLARE_SWAP_DERIVED(X) void swap(X& other) { doSwap(other); }
- #define KDCHART_DECLARE_SWAP_SPECIALISATION(X)
- #define KDCHART_DECLARE_SWAP_SPECIALISATION_DERIVED(X) KDCHART_DECLARE_SWAP_SPECIALISATION(X)
- #define KDCHART_DERIVED_PRIVATE_FOOTER(CLASS, PARENT)
- #define KDCHART_IMPL_DERIVED_DIAGRAM(CLASS, PARENT, PLANE)
- #define KDCHART_IMPL_DERIVED_PLANE(CLASS, BASEPLANE)

Functions

- template<typename T> T & __kdab__dereference_for_methodcall (T ∗o)
- template<typename T> T & __kdab__dereference_for_methodcall (T &o)

8.48.1 Define Documentation

8.48.1.1 #define KDAB_SET_OBJECT_NAME(x) __kdab__dereference_for_methodcall(x).setObjectName(QLatin1String(#x))

Definition at line 46 of file KDChartGlobal.h.

8.48.1.2 #define KDCHART_DECLARE_DERIVED_DIAGRAM(X, PLANE)

Value:

```
protected:                                              \
    class Private;                                      \
    inline Private * d_func();                          \
    inline const Private * d_func() const;              \
    explicit inline X( Private * );                     \
    explicit inline X( Private *, QWidget *, PLANE * ); \
private:                                                \
    void init();
```

Definition at line 173 of file KDChartGlobal.h.

8.48.1.3 #define KDCHART_DECLARE_PRIVATE_BASE_POLYMORPHIC(X)

Value:

```
protected:                                        \
    class Private;                                    \
    Private * d_func() { return _d; }                 \
    const Private * d_func() const { return _d; }     \
    explicit inline X( Private * );                   \
private:                                              \
    void init();                                      \
    Private * _d;
```

Definition at line 117 of file KDChartGlobal.h.

8.48.1.4 #define KDCHART_DECLARE_PRIVATE_BASE_POLYMORPHIC_QWIDGET(X)

Value:

```
protected:                                        \
    class Private;                                    \
    Private * d_func() { return _d; }                 \
    const Private * d_func() const { return _d; }     \
    explicit inline X( Private *, QWidget* );                  \
private:                                              \
    void init();                                      \
    Private * _d;
```

Definition at line 140 of file KDChartGlobal.h.

8.48.1.5 #define KDCHART_DECLARE_PRIVATE_BASE_VALUE(X)

Value:

```
public:                                                      \
    inline void swap( X & other ) { qSwap( _d, other._d ); } \
protected:                                                   \
    class Private;                                           \
    Private * d_func() { return _d; }                        \
    const Private * d_func() const { return _d; }            \
private:                                                     \
    void init();                                             \
    Private * _d;
```

Definition at line 94 of file KDChartGlobal.h.

8.48.1.6 #define KDCHART_DECLARE_PRIVATE_DERIVED(X)

Value:

```
protected:                                          \
    class Private;                                  \
    inline Private * d_func();                      \
    inline const Private * d_func() const;          \
    explicit inline X( Private * );                 \
private:                                            \
    void init();
```

Definition at line 60 of file KDChartGlobal.h.

8.48.1.7 #define KDCHART_DECLARE_PRIVATE_DERIVED_PARENT(X, ParentType)

Value:

```
protected:                                          \
    class Private;                                  \
    inline Private * d_func();                      \
    inline const Private * d_func() const;          \
    explicit inline X( Private *, ParentType );     \
private:                                            \
    void init();
```

Definition at line 81 of file KDChartGlobal.h.

8.48.1.8 #define KDCHART_DECLARE_PRIVATE_DERIVED_QWIDGET(X) KDCHART_DECLARE_PRIVATE_DERIVED_PARENT(X, QWidget*)

Definition at line 91 of file KDChartGlobal.h.

8.48.1.9 #define KDCHART_DECLARE_SWAP_BASE(X)

Value:

```
protected: \
    void doSwap( X& other ) \
    { qSwap( _d, other._d); }
```

Definition at line 224 of file KDChartGlobal.h.

8.48.1.10 #define KDCHART_DECLARE_SWAP_DERIVED(X) void swap(X& other) { doSwap(other); }

Definition at line 229 of file KDChartGlobal.h.

8.48.1.11 #define KDCHART_DECLARE_SWAP_SPECIALISATION(X)

Value:

```
template <> inline void qSwap<X>( X & lhs, X & rhs )    \
    { lhs.swap( rhs ); }                                    \
    namespace std {                                         \
        template <> inline void swap<X>( X & lhs, X & rhs ) \
        { lhs.swap( rhs ); }                                \
    }
```

Definition at line 208 of file KDChartGlobal.h.

8.48.1.12 #define KDCHART_DECLARE_SWAP_SPECIALISATION_DERIVED(X) KDCHART_DECLARE_SWAP_SPECIALISATION(X)

Definition at line 221 of file KDChartGlobal.h.

8.48.1.13 #define KDCHART_DERIVED_PRIVATE_FOOTER(CLASS, PARENT)

Value:

```
inline CLASS::CLASS( Private * p )                          \
  : PARENT( p ) { init(); }                                 \
inline CLASS::Private * CLASS::d_func()                     \
{ return static_cast<Private*>( PARENT::d_func() ); }       \
inline const CLASS::Private * CLASS::d_func() const         \
{ return static_cast<const Private*>( PARENT::d_func() ); }
```

Definition at line 152 of file KDChartGlobal.h.

8.48.1.14 #define KDCHART_IMPL_DERIVED_DIAGRAM(CLASS, PARENT, PLANE)

Value:

```
inline CLASS::CLASS( Private * p )                           \
    : PARENT( p ) { init(); }                                \
inline CLASS::CLASS(                                         \
    Private * p, QWidget* parent, PLANE * plane )            \
    : PARENT( p, parent, plane ) { init(); }                 \
inline CLASS::Private * CLASS::d_func()                      \
    { return static_cast<Private *>( PARENT::d_func() ); }   \
inline const CLASS::Private * CLASS::d_func() const          \
    { return static_cast<const Private *>( PARENT::d_func() ); }
```

Definition at line 184 of file KDChartGlobal.h.

8.48.1.15 #define KDCHART_IMPL_DERIVED_PLANE(CLASS, BASEPLANE)

Value:

```
inline CLASS::CLASS( Private * p, Chart* parent )           \
    : BASEPLANE( p, parent ) { init(); }                      \
inline CLASS::Private * CLASS::d_func()                       \
    { return static_cast<Private *>( BASEPLANE::d_func() ); } \
inline const CLASS::Private * CLASS::d_func() const           \
    { return static_cast<const Private *>( BASEPLANE::d_func() ); }
```

Definition at line 196 of file KDChartGlobal.h.

8.48.2 Function Documentation

8.48.2.1 template<typename T> T& __kdab__dereference_for_methodcall (T * *o*)

Definition at line 42 of file KDChartGlobal.h.

```
42                                                          {
43       return *o;
44 }
```

8.48.2.2 template<typename T> T& __kdab__dereference_for_methodcall (T & *o*)

Definition at line 37 of file KDChartGlobal.h.

```
37                                                          {
38       return o;
39 }
```

8.49 KDChartGridAttributes.cpp File Reference

`#include "KDChartGridAttributes.h"`

`#include <QPen>`

`#include <QDebug>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartGridAttributes.cpp:

Defines

- #define d d_func()

Functions

- QDebug operator<< (QDebug dbg, const KDChart::GridAttributes &a)

8.49.1 Define Documentation

8.49.1.1 #define d d_func()

Definition at line 33 of file KDChartGridAttributes.cpp.

8.49.2 Function Documentation

8.49.2.1 QDebug operator<< (QDebug *dbg*, const KDChart::GridAttributes & *a*)

Definition at line 279 of file KDChartGridAttributes.cpp.

```
{
    dbg << "KDChart::GridAttributes("
            << "visible="<<a.isGridVisible()
            << "subVisible="<<a.isSubGridVisible()
            // KDChartEnums::GranularitySequence sequence;
            << "stepWidth=" << a.gridStepWidth()
            << "subStepWidth=" << a.gridSubStepWidth()
            << "pen="<<a.gridPen()
            << "subPen="<<a.subGridPen()
            << "zeroPen="<<a.zeroLinePen()
            << ")";
    return dbg;
}
```

8.50 KDChartGridAttributes.h File Reference

`#include <QMetaType>`

`#include "KDChartGlobal.h"`

`#include "KDChartEnums.h"`

Include dependency graph for KDChartGridAttributes.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

Functions

- KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::GridAttributes &)
- Q_DECLARE_TYPEINFO (KDChart::GridAttributes, Q_MOVABLE_TYPE)

8.50.1 Function Documentation

8.50.1.1 KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::GridAttributes &)

Definition at line 279 of file KDChartGridAttributes.cpp.

References KDChart::GridAttributes::gridPen(), KDChart::GridAttributes::gridStepWidth(), KDChart::GridAttributes::gridSubStepWidth(), KDChart::GridAttributes::isGridVisible(), KDChart::GridAttributes::isSubGridVisible(), KDChart::GridAttributes::subGridPen(), and KDChart::GridAttributes::zeroLinePen().

```
{
    dbg << "KDChart::GridAttributes("
            << "visible="<<a.isGridVisible()
            << "subVisible="<<a.isSubGridVisible()
            // KDChartEnums::GranularitySequence sequence;
            << "stepWidth=" << a.gridStepWidth()
            << "subStepWidth=" << a.gridSubStepWidth()
            << "pen="<<a.gridPen()
            << "subPen="<<a.subGridPen()
            << "zeroPen="<<a.zeroLinePen()
            << ")";
    return dbg;
}
```

8.50.1.2 Q_DECLARE_TYPEINFO (KDChart::GridAttributes, Q_MOVABLE_TYPE)

8.51 KDChartHeaderFooter.cpp File Reference

```
#include "KDChartChart.h"
#include "KDChartHeaderFooter.h"
#include "KDChartHeaderFooter_p.h"
#include <KDChartTextAttributes.h>
#include <QFont>
#include <QPainter>
#include <QAbstractTextDocumentLayout>
#include <QTextDocumentFragment>
#include <QTextBlock>
#include <QtDebug>
#include <QLabel>
#include "KDTextDocument.h"
#include <KDABLibFakes>
```

Include dependency graph for KDChartHeaderFooter.cpp:

Defines

- #define d d_func()

8.51.1 Define Documentation

8.51.1.1 #define d d_func()

Definition at line 53 of file KDChartHeaderFooter.cpp.

8.52 KDChartHeaderFooter.h File Reference

`#include "KDChartTextArea.h"`

`#include "KDChartPosition.h"`

Include dependency graph for KDChartHeaderFooter.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.53 KDChartLayoutItems.cpp File Reference

```
#include "KDChartLayoutItems.h"
#include "KDTextDocument.h"
#include "KDChartAbstractArea.h"
#include "KDChartAbstractDiagram.h"
#include "KDChartBackgroundAttributes.h"
#include "KDChartFrameAttributes.h"
#include "KDChartPaintContext.h"
#include "KDChartPainterSaver_p.h"
#include <QTextCursor>
#include <QTextBlockFormat>
#include <QTextDocumentFragment>
#include <QAbstractTextDocumentLayout>
#include <QLayout>
#include <QPainter>
#include <QDebug>
#include <QCoreApplication>
#include <QApplication>
#include <QStringList>
#include <QStyle>
#include <KDABLibFakes>
#include <math.h>
```

Include dependency graph for KDChartLayoutItems.cpp:

Defines

- #define PI 3.141592653589793

Functions

- QPointF rotatedPoint (const QPointF &pt, qreal rotation)
- QRectF rotatedRect (const QRectF &rect, qreal angle)
- void updateCommonBrush (QBrush &commonBrush, bool &bStart, const KDChart::AbstractArea &area)

8.53.1 Define Documentation

8.53.1.1 #define PI 3.141592653589793

Definition at line 50 of file KDChartLayoutItems.cpp.

Referenced by KDChart::TextLayoutItem::intersects(), and rotatedPoint().

8.53.2 Function Documentation

8.53.2.1 QPointF rotatedPoint (const QPointF & *pt*, qreal *rotation*) `[static]`

Definition at line 357 of file KDChartLayoutItems.cpp.

References PI.

Referenced by rotatedRect().

```
{
    const qreal angle = PI * rotation / 180.0;
    const qreal cosAngle = cos( angle );
    const qreal sinAngle = sin( angle );
    return QPointF(
            (cosAngle * pt.x() + sinAngle * pt.y() ),
            (cosAngle * pt.y() + sinAngle * pt.x() ) );
}
```

8.53.2.2 QRectF rotatedRect (const QRectF & *rect*, qreal *angle*) `[static]`

Definition at line 367 of file KDChartLayoutItems.cpp.

References rotatedPoint().

Referenced by KDChart::TextLayoutItem::paint().

```
{
    const QPointF topLeft(  rotatedPoint( rect.topLeft(),  angle ) );
    //const QPointF topRight( rotatedPoint( rect.topRight(), angle ) );
    //const QPointF bottomLeft(  rotatedPoint( rect.bottomLeft(),  angle ) );
    //const QPointF bottomRight( rotatedPoint( rect.bottomRight(), angle ) );
    const QPointF siz( rotatedPoint( QPointF( rect.size().width(), rect.size().height() ), angle ) );
    const QRectF result(
            topLeft,
            QSizeF( siz.x(), //bottomRight.x() - topLeft.x(),
                    siz.y() ) ); //bottomRight.y() - topLeft.y() ) );
    //qDebug() << "angle" << angle << "\nbefore:" << rect << "\n after:" << result;
    return result;
}
```

8.53.2.3 void updateCommonBrush (QBrush & *commonBrush*, bool & *bStart*, const KDChart::AbstractArea & *area*) `[static]`

Definition at line 798 of file KDChartLayoutItems.cpp.

References KDChart::AbstractAreaBase::backgroundAttributes(), KDChart::BackgroundAttributes::brush(), KDChart::AbstractAreaBase::frameAttributes(), KDChart::BackgroundAttributes::isVisible(), KDChart::FrameAttributes::isVisible(), and KDChart::BackgroundAttributes::pixmapMode().

Referenced by KDChart::AutoSpacerLayoutItem::sizeHint().

```
{
    const KDChart::BackgroundAttributes ba( area.backgroundAttributes() );
    const bool hasSimpleBrush = (
            ! area.frameAttributes().isVisible() &&
            ba.isVisible() &&
            ba.pixmapMode() == KDChart::BackgroundAttributes::BackgroundPixmapModeNone &&
            ba.brush().gradient() == 0 );
    if( bStart ){
        bStart = false;
        commonBrush = hasSimpleBrush ? ba.brush() : QBrush();
    }else{
        if( ! hasSimpleBrush || ba.brush() != commonBrush )
        {
            commonBrush = QBrush();
        }
    }
}
```

8.54 KDChartLayoutItems.h File Reference

```
#include <QBrush>
#include <QFont>
#include <QFontMetricsF>
#include <QLayout>
#include <QLayoutItem>
#include <QPen>
#include "KDChartTextAttributes.h"
#include "KDChartMarkerAttributes.h"
```

Include dependency graph for KDChartLayoutItems.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.55 KDChartLegend.cpp File Reference

```
#include "KDChartLegend.h"
#include "KDChartLegend_p.h"
#include <KDChartTextAttributes.h>
#include <KDChartMarkerAttributes.h>
#include <QFont>
#include <QPainter>
#include <QTextTableCell>
#include <QTextCursor>
#include <QTextCharFormat>
#include <QTextDocumentFragment>
#include <QTimer>
#include <QAbstractTextDocumentLayout>
#include <QtDebug>
#include <QLabel>
#include <KDChartAbstractDiagram.h>
#include "KDTextDocument.h"
#include <KDChartDiagramObserver.h>
#include <QGridLayout>
#include "KDChartLayoutItems.h"
#include <KDABLibFakes>
```

Include dependency graph for KDChartLegend.cpp:

Defines

- #define d d_func()

8.55.1 Define Documentation

8.55.1.1 #define d d_func()

Definition at line 82 of file KDChartLegend.cpp.

8.56 KDChartLegend.h File Reference

`#include "KDChartAbstractAreaWidget.h"`

`#include "KDChartPosition.h"`

`#include "KDChartMarkerAttributes.h"`

Include dependency graph for KDChartLegend.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.57 KDChartLineAttributes.cpp File Reference

`#include "KDChartLineAttributes.h"`

`#include <QDebug>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartLineAttributes.cpp:

Defines

- #define d d_func()

Functions

- QDebug operator<< (QDebug dbg, const KDChart::LineAttributes &a)

8.57.1 Define Documentation

8.57.1.1 #define d d_func()

Definition at line 31 of file KDChartLineAttributes.cpp.

8.57.2 Function Documentation

8.57.2.1 QDebug operator<< (QDebug *dbg*, const KDChart::LineAttributes & *a*)

Definition at line 123 of file KDChartLineAttributes.cpp.

```
{
    dbg << "KDChart::LineAttributes("
            //     MissingValuesPolicy missingValuesPolicy;
            << "bool="<<a.displayArea()
            << "transparency="<<a.transparency()
            << ")";
    return dbg;

}
```

8.58 KDChartLineAttributes.h File Reference

`#include <QMetaType>`

`#include "KDChartGlobal.h"`

Include dependency graph for KDChartLineAttributes.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

Functions

- KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::LineAttributes &)
- Q_DECLARE_TYPEINFO (KDChart::LineAttributes, Q_MOVABLE_TYPE)

8.58.1 Function Documentation

8.58.1.1 KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::LineAttributes &)

Definition at line 123 of file KDChartLineAttributes.cpp.

References KDChart::LineAttributes::displayArea(), and KDChart::LineAttributes::transparency().

```
{
    dbg << "KDChart::LineAttributes("
            //     MissingValuesPolicy missingValuesPolicy;
            << "bool="<<a.displayArea()
            << "transparency="<<a.transparency()
            << ")";
    return dbg;

}
```

8.58.1.2 Q_DECLARE_TYPEINFO (KDChart::LineAttributes, Q_MOVABLE_TYPE)

8.59 KDChartLineDiagram.cpp File Reference

```
#include <QDebug>
#include <QPainter>
#include <QString>
#include <QPainterPath>
#include <QPen>
#include <QVector>
#include "KDChartLineDiagram.h"
#include "KDChartLineDiagram_p.h"
#include "KDChartBarDiagram.h"
#include "KDChartPalette.h"
#include "KDChartPosition.h"
#include "KDChartTextAttributes.h"
#include "KDChartThreeDLineAttributes.h"
#include "KDChartAttributesModel.h"
#include "KDChartAbstractGrid.h"
#include "KDChartDataValueAttributes.h"
#include <KDABLibFakes>
```

Include dependency graph for KDChartLineDiagram.cpp:

Defines

- #define d d_func()

8.59.1 Define Documentation

8.59.1.1 #define d d_func()

Definition at line 57 of file KDChartLineDiagram.cpp.

8.60 KDChartLineDiagram.h File Reference

`#include "KDChartAbstractCartesianDiagram.h"`

`#include "KDChartLineAttributes.h"`

Include dependency graph for KDChartLineDiagram.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.61 KDChartMarkerAttributes.cpp File Reference

`#include "KDChartMarkerAttributes.h"`

`#include <QColor>`

`#include <QMap>`

`#include <QPen>`

`#include <QSizeF>`

`#include <QDebug>`

`#include <qglobal.h>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartMarkerAttributes.cpp:

Defines

- #define d d_func()

Functions

- QDebug operator<< (QDebug dbg, const MarkerAttributes &ma)

8.61.1 Define Documentation

8.61.1.1 #define d d_func()

Definition at line 85 of file KDChartMarkerAttributes.cpp.

8.61.2 Function Documentation

8.61.2.1 QDebug operator<< (QDebug *dbg*, const MarkerAttributes & *ma*)

Definition at line 172 of file KDChartMarkerAttributes.cpp.

References KDChart::MarkerAttributes::isVisible(), KDChart::MarkerAttributes::markerColor(), KDChart::MarkerAttributes::markerStyle(), KDChart::MarkerAttributes::markerStylesMap(), and KDChart::MarkerAttributes::pen().

```
172                                                              {
173     return dbg << "KDChart::MarkerAttributes("
174                << "visible=" << ma.isVisible()
175                << "markerStylesMap=" << ma.markerStylesMap()
176                << "markerStyle=" << ma.markerStyle()
177                << "markerColor=" << ma.markerColor()
178                << "pen=" << ma.pen()
179                << ")";
180 }
```

8.62 KDChartMarkerAttributes.h File Reference

`#include <QMetaType>`

`#include "KDChartGlobal.h"`

Include dependency graph for KDChartMarkerAttributes.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

Functions

- KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::MarkerAttributes &)
- Q_DECLARE_TYPEINFO (KDChart::MarkerAttributes, Q_MOVABLE_TYPE)

8.62.1 Function Documentation

8.62.1.1 KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::MarkerAttributes &)

8.62.1.2 Q_DECLARE_TYPEINFO (KDChart::MarkerAttributes, Q_MOVABLE_TYPE)

8.63 KDChartMeasure.cpp File Reference

`#include <QWidget>`

`#include "KDChartMeasure.h"`

`#include <QtXml/QDomDocumentFragment>`

`#include <KDChartAbstractArea.h>`

`#include <KDChartTextAttributes.h>`

`#include <KDChartFrameAttributes.h>`

`#include <KDChartBackgroundAttributes.h>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartMeasure.cpp:

Namespaces

- namespace KDChart

Functions

- QDebug operator<< (QDebug dbg, const KDChart::Measure &m)

8.63.1 Function Documentation

8.63.1.1 QDebug operator<< (QDebug *dbg*, const KDChart::Measure & *m*)

Definition at line 226 of file KDChartMeasure.cpp.

```
{
    dbg << "KDChart::Measure("
        << "value="<<m.value()
        << "calculationmode="<<m.calculationMode()
        << "referencearea="<<m.referenceArea()
        << "referenceorientation="<<m.referenceOrientation()
        << ")";
    return dbg;
}
```

8.64 KDChartMeasure.h File Reference

8.64.1 Detailed Description

Declaring the class KDChart::Measure.

Definition in file KDChartMeasure.h.

`#include <QDebug>`

`#include <Qt>`

`#include <QStack>`

`#include "KDChartGlobal.h"`

`#include "KDChartEnums.h"`

Include dependency graph for KDChartMeasure.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

Functions

- KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::Measure &)

8.64.2 Function Documentation

8.64.2.1 KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::Measure &)

Definition at line 226 of file KDChartMeasure.cpp.

References KDChart::Measure::calculationMode(), KDChart::Measure::referenceArea(), KDChart::Measure::referenceOrientation(), and KDChart::Measure::value().

```
227 {
228     dbg << "KDChart::Measure("
229         << "value="<<m.value()
230         << "calculationmode="<<m.calculationMode()
231         << "referencearea="<<m.referenceArea()
232         << "referenceorientation="<<m.referenceOrientation()
233         << ")";
234     return dbg;
235 }
```

8.65 KDChartPaintContext.cpp File Reference

`#include <QRectF>`

`#include <QPainter>`

`#include "KDChartPaintContext.h"`

`#include "KDChartAbstractCoordinatePlane.h"`

`#include <KDABLibFakes>`

Include dependency graph for KDChartPaintContext.cpp:

Defines

- #define d (d_func())

8.65.1 Define Documentation

8.65.1.1 #define d (d_func())

Definition at line 36 of file KDChartPaintContext.cpp.

8.66 KDChartPaintContext.h File Reference

`#include <QRectF>`

`#include "KDChartGlobal.h"`

Include dependency graph for KDChartPaintContext.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.67 KDChartPalette.cpp File Reference

`#include "KDChartPalette.h"`

`#include <QBrush>`

`#include <QVector>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartPalette.cpp:

Defines

- #define d d_func()

Functions

- Palette makeDefaultPalette ()
- Palette makeRainbowPalette ()
- Palette makeSubduedPalette ()

8.67.1 Define Documentation

8.67.1.1 #define d d_func()

Definition at line 103 of file KDChartPalette.cpp.

8.67.2 Function Documentation

8.67.2.1 Palette makeDefaultPalette () `[static]`

Definition at line 40 of file KDChartPalette.cpp.

References KDChart::Palette::addBrush().

```
40                                      {
41         Palette p;
42
43         p.addBrush( Qt::red );
44         p.addBrush( Qt::green );
45         p.addBrush( Qt::blue );
46         p.addBrush( Qt::cyan );
47         p.addBrush( Qt::magenta );
48         p.addBrush( Qt::yellow );
49         p.addBrush( Qt::darkRed );
50         p.addBrush( Qt::darkGreen );
51         p.addBrush( Qt::darkBlue );
52         p.addBrush( Qt::darkCyan );
53         p.addBrush( Qt::darkMagenta );
54         p.addBrush( Qt::darkYellow );
55
56         return p;
57     }
```

8.67.2.2 Palette makeRainbowPalette () `[static]`

Definition at line 84 of file KDChartPalette.cpp.

References KDChart::Palette::addBrush(), and KDChart::Palette::getBrush().

```
84                                         {
85         Palette p;
86 
87         p.addBrush( QColor(255,  0,196) );
88         p.addBrush( QColor(255,  0, 96) );
89         p.addBrush( QColor(255, 128,64) );
90         p.addBrush( Qt::yellow );
91         p.addBrush( Qt::green );
92         p.addBrush( Qt::cyan );
93         p.addBrush( QColor( 96, 96,255) );
94         p.addBrush( QColor(160,  0,255) );
95         for( int i = 8 ; i < 16 ; ++i )
96             p.addBrush( p.getBrush(i-8).color().light(), i );
97 
98         return p;
99     }
```

8.67.2.3 Palette makeSubduedPalette () `[static]`

Definition at line 59 of file KDChartPalette.cpp.

References KDChart::Palette::addBrush().

```
59                                         {
60         Palette p;
61 
62         p.addBrush( QColor( 0xe0,0x7f,0x70 ) );
63         p.addBrush( QColor( 0xe2,0xa5,0x6f ) );
64         p.addBrush( QColor( 0xe0,0xc9,0x70 ) );
65         p.addBrush( QColor( 0xd1,0xe0,0x70 ) );
66         p.addBrush( QColor( 0xac,0xe0,0x70 ) );
67         p.addBrush( QColor( 0x86,0xe0,0x70 ) );
68         p.addBrush( QColor( 0x70,0xe0,0x7f ) );
69         p.addBrush( QColor( 0x70,0xe0,0xa4 ) );
70         p.addBrush( QColor( 0x70,0xe0,0xc9 ) );
71         p.addBrush( QColor( 0x70,0xd1,0xe0 ) );
72         p.addBrush( QColor( 0x70,0xac,0xe0 ) );
73         p.addBrush( QColor( 0x70,0x86,0xe0 ) );
74         p.addBrush( QColor( 0x7f,0x70,0xe0 ) );
75         p.addBrush( QColor( 0xa4,0x70,0xe0 ) );
76         p.addBrush( QColor( 0xc9,0x70,0xe0 ) );
77         p.addBrush( QColor( 0xe0,0x70,0xd1 ) );
78         p.addBrush( QColor( 0xe0,0x70,0xac ) );
79         p.addBrush( QColor( 0xe0,0x70,0x86 ) );
80 
81         return p;
82     }
```

8.68 KDChartPalette.h File Reference

`#include <QObject>`

`#include <QBrush>`

`#include "KDChartGlobal.h"`

Include dependency graph for KDChartPalette.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.69 KDChartPieAttributes.cpp File Reference

`#include "KDChartPieAttributes.h"`

`#include "KDChartPieAttributes_p.h"`

`#include <QDebug>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartPieAttributes.cpp:

Defines

- #define d d_func()

Functions

- QDebug operator<< (QDebug dbg, const KDChart::PieAttributes &a)

8.69.1 Define Documentation

8.69.1.1 #define d d_func()

Definition at line 33 of file KDChartPieAttributes.cpp.

8.69.2 Function Documentation

8.69.2.1 QDebug operator<< (QDebug *dbg*, const KDChart::PieAttributes & *a*)

Definition at line 106 of file KDChartPieAttributes.cpp.

```
{
    dbg << "KDChart::PieAttributes(";
    dbg << "explodeFactor="<< a.explodeFactor() << ")";
    return dbg;
}
```

8.70 KDChartPieAttributes.h File Reference

`#include <QMetaType>`

`#include "KDChartAbstractThreeDAttributes.h"`

`#include "KDChartGlobal.h"`

Include dependency graph for KDChartPieAttributes.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

Functions

- KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::PieAttributes &)
- Q_DECLARE_TYPEINFO (KDChart::PieAttributes, Q_MOVABLE_TYPE)

8.70.1 Function Documentation

8.70.1.1 KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::PieAttributes &)

Definition at line 106 of file KDChartPieAttributes.cpp.

References KDChart::PieAttributes::explodeFactor().

```
107 {
108     dbg << "KDChart::PieAttributes(";
109     dbg << "explodeFactor="<< a.explodeFactor() << ")";
110     return dbg;
111 }
```

8.70.1.2 Q_DECLARE_TYPEINFO (KDChart::PieAttributes, Q_MOVABLE_TYPE)

8.71 KDChartPieDiagram.cpp File Reference

`#include <QDebug>`

`#include <QPainter>`

`#include <QStack>`

`#include "KDChartAttributesModel.h"`

`#include "KDChartPaintContext.h"`

`#include "KDChartPieDiagram.h"`

`#include "KDChartPieDiagram_p.h"`

`#include "KDChartPieAttributes.h"`

`#include "KDChartThreeDPieAttributes.h"`

`#include "KDChartPainterSaver_p.h"`

`#include "KDChartDataValueAttributes.h"`

`#include <KDABLibFakes>`

Include dependency graph for KDChartPieDiagram.cpp:

Defines

- #define d d_func()

Functions

- QRectF buildReferenceRect (const PolarCoordinatePlane ∗plane)

8.71.1 Define Documentation

8.71.1.1 #define d d_func()

Definition at line 50 of file KDChartPieDiagram.cpp.

8.71.2 Function Documentation

8.71.2.1 QRectF buildReferenceRect (const PolarCoordinatePlane ∗ *plane*) `[static]`

Definition at line 113 of file KDChartPieDiagram.cpp.

References KDChart::PolarCoordinatePlane::translate().

Referenced by KDChart::PieDiagram::paint().

```
{
    QRectF contentsRect;
//qDebug() << "..........................................";
    QPointF referencePointAtTop = plane->translate( QPointF( 1, 0 ) );
    QPointF temp = plane->translate( QPointF( 0, 0 ) ) - referencePointAtTop;
    const double offset = temp.y();
    referencePointAtTop.setX( referencePointAtTop.x() - offset );
    contentsRect.setTopLeft( referencePointAtTop );
    contentsRect.setBottomRight( referencePointAtTop + QPointF( 2*offset, 2*offset) );
//qDebug() << contentsRect;
    return contentsRect;
}
```

8.72 KDChartPieDiagram.h File Reference

`#include "KDChartAbstractPieDiagram.h"`

Include dependency graph for KDChartPieDiagram.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.73 KDChartPolarCoordinatePlane.cpp File Reference

```
#include <math.h>
#include <QFont>
#include <QList>
#include <QtDebug>
#include <QPainter>
#include "KDChartChart.h"
#include "KDChartPaintContext.h"
#include "KDChartAbstractDiagram.h"
#include "KDChartAbstractPolarDiagram.h"
#include "KDChartPolarCoordinatePlane.h"
#include "KDChartPolarCoordinatePlane_p.h"
#include "KDChartPainterSaver_p.h"
#include <KDABLibFakes>
```

Include dependency graph for KDChartPolarCoordinatePlane.cpp:

Defines

- #define d d_func()

8.73.1 Define Documentation

8.73.1.1 #define d d_func()

Definition at line 45 of file KDChartPolarCoordinatePlane.cpp.

8.74 KDChartPolarCoordinatePlane.h File Reference

`#include "KDChartAbstractCoordinatePlane.h"`

Include dependency graph for KDChartPolarCoordinatePlane.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.75 KDChartPolarDiagram.cpp File Reference

```
#include <QPainter>
#include "KDChartAttributesModel.h"
#include "KDChartPaintContext.h"
#include "KDChartPolarDiagram.h"
#include "KDChartPolarDiagram_p.h"
#include "KDChartPainterSaver_p.h"
#include "KDChartDataValueAttributes.h"
#include <KDABLibFakes>
```

Include dependency graph for KDChartPolarDiagram.cpp:

Defines

- #define d d_func()

8.75.1 Define Documentation

8.75.1.1 #define d d_func()

Definition at line 47 of file KDChartPolarDiagram.cpp.

8.76 KDChartPolarDiagram.h File Reference

`#include "KDChartPosition.h"`

`#include "KDChartAbstractPolarDiagram.h"`

Include dependency graph for KDChartPolarDiagram.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.77 KDChartPosition.cpp File Reference

`#include <KDChartPosition.h>`

`#include <KDChartEnums.h>`

`#include <QString>`

`#include <QStringList>`

`#include <QList>`

`#include <QByteArray>`

`#include <KDABLibFakes>`

`#include <cassert>`

Include dependency graph for KDChartPosition.cpp:

Functions

- QDebug operator<< (QDebug dbg, const KDChart::Position &p)

Variables

- int maxPositionValue = 10
- Position staticPositionCenter = Position(KDChartEnums::PositionCenter)
- Position staticPositionEast = Position(KDChartEnums::PositionEast)
- Position staticPositionFloating = Position(KDChartEnums::PositionFloating)
- const char ∗ staticPositionNames [ ]
- Position staticPositionNorth = Position(KDChartEnums::PositionNorth)
- Position staticPositionNorthEast = Position(KDChartEnums::PositionNorthEast)
- Position staticPositionNorthWest = Position(KDChartEnums::PositionNorthWest)
- Position staticPositionSouth = Position(KDChartEnums::PositionSouth)
- Position staticPositionSouthEast = Position(KDChartEnums::PositionSouthEast)
- Position staticPositionSouthWest = Position(KDChartEnums::PositionSouthWest)
- Position staticPositionUnknown = Position(KDChartEnums::PositionUnknown)
- Position staticPositionWest = Position(KDChartEnums::PositionWest)

8.77.1 Function Documentation

8.77.1.1 QDebug operator<< (QDebug *dbg*, const KDChart::Position & *p*)

Definition at line 260 of file KDChartPosition.cpp.

```
{
    dbg << "KDChart::Position("
        << p.name() << ")";
    return dbg;
}
```

8.77.2 Variable Documentation

8.77.2.1 int maxPositionValue = 10 `[static]`

Definition at line 80 of file KDChartPosition.cpp.

Referenced by KDChart::Position::fromName(), KDChart::Position::names(), and KDChart::Position::printableNames().

8.77.2.2 Position staticPositionCenter = Position(KDChartEnums::PositionCenter) `[static]`

Definition at line 69 of file KDChartPosition.cpp.

8.77.2.3 Position staticPositionEast = Position(KDChartEnums::PositionEast) `[static]`

Definition at line 73 of file KDChartPosition.cpp.

8.77.2.4 Position staticPositionFloating = Position(KDChartEnums::PositionFloating) `[static]`

Definition at line 78 of file KDChartPosition.cpp.

8.77.2.5 const char∗ staticPositionNames[ ] `[static]`

Initial value:

```
 {
   QT_TRANSLATE_NOOP("Position","Unknown Position"),
   QT_TRANSLATE_NOOP("Position","Center"),
   QT_TRANSLATE_NOOP("Position","NorthWest"),
   QT_TRANSLATE_NOOP("Position","North"),
   QT_TRANSLATE_NOOP("Position","NorthEast"),
   QT_TRANSLATE_NOOP("Position","East"),
   QT_TRANSLATE_NOOP("Position","SouthEast"),
   QT_TRANSLATE_NOOP("Position","South"),
   QT_TRANSLATE_NOOP("Position","SouthWest"),
   QT_TRANSLATE_NOOP("Position","West"),

}
```

Definition at line 49 of file KDChartPosition.cpp.

Referenced by KDChart::Position::fromName(), KDChart::Position::name(), KDChart::Position::names(), and KDChart::Position::printableName().

8.77.2.6 Position staticPositionNorth = Position(KDChartEnums::PositionNorth) `[static]`

Definition at line 71 of file KDChartPosition.cpp.

8.77.2.7 Position staticPositionNorthEast = Position(KDChartEnums::PositionNorthEast) `[static]`

Definition at line 72 of file KDChartPosition.cpp.

8.77.2.8 Position staticPositionNorthWest = Position(KDChartEnums::PositionNorthWest) `[static]`

Definition at line 70 of file KDChartPosition.cpp.

8.77.2.9 Position staticPositionSouth = Position(KDChartEnums::PositionSouth) `[static]`

Definition at line 75 of file KDChartPosition.cpp.

8.77.2.10 Position staticPositionSouthEast = Position(KDChartEnums::PositionSouthEast) `[static]`

Definition at line 74 of file KDChartPosition.cpp.

8.77.2.11 Position staticPositionSouthWest = Position(KDChartEnums::PositionSouthWest) `[static]`

Definition at line 76 of file KDChartPosition.cpp.

8.77.2.12 Position staticPositionUnknown = Position(KDChartEnums::PositionUnknown) `[static]`

Definition at line 68 of file KDChartPosition.cpp.

8.77.2.13 Position staticPositionWest = Position(KDChartEnums::PositionWest) `[static]`

Definition at line 77 of file KDChartPosition.cpp.

8.78 KDChartPosition.h File Reference

```
#include <QDebug>
#include <Qt>
#include <QMetaType>
#include <QCoreApplication>
#include "KDChartGlobal.h"
#include "KDChartEnums.h"
```

Include dependency graph for KDChartPosition.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

Functions

- KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::Position &)
- Q_DECLARE_TYPEINFO (KDChart::Position, Q_MOVABLE_TYPE)

8.78.1 Function Documentation

8.78.1.1 KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::Position &)

Definition at line 260 of file KDChartPosition.cpp.

References KDChart::Position::name().

```
{
    dbg << "KDChart::Position("
        << p.name() << ")";
    return dbg;
}
```

8.78.1.2 Q_DECLARE_TYPEINFO (KDChart::Position, Q_MOVABLE_TYPE)

8.79 KDChartRelativePosition.cpp File Reference

`#include "KDChartRelativePosition.h"`

`#include "KDChartEnums.h"`

`#include "KDChartMeasure.h"`

`#include "KDChartPosition.h"`

`#include "KDChartAbstractArea.h"`

`#include <QWidget>`

`#include <QLayout>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartRelativePosition.cpp:

Defines

- #define d d_func()

Functions

- QDebug operator<< (QDebug dbg, const KDChart::RelativePosition &rp)

8.79.1 Define Documentation

8.79.1.1 #define d d_func()

Definition at line 93 of file KDChartRelativePosition.cpp.

8.79.2 Function Documentation

8.79.2.1 QDebug operator<< (QDebug *dbg*, const KDChart::RelativePosition & *rp*)

Definition at line 210 of file KDChartRelativePosition.cpp.

```
{
    dbg << "KDChart::RelativePosition("
        << "referencearea="<<rp.referenceArea()
        << "referenceposition="<<rp.referencePosition()
        << "alignment="<<rp.alignment()
        << "horizontalpadding="<<rp.horizontalPadding()
        << "verticalpadding="<<rp.verticalPadding()
        << "rotation="<<rp.rotation()
        << ")";
    return dbg;
}
```

8.80 KDChartRelativePosition.h File Reference

`#include <QDebug>`

`#include <QMetaType>`

`#include <Qt>`

`#include <QPointF>`

`#include <QSizeF>`

`#include "KDChartGlobal.h"`

Include dependency graph for KDChartRelativePosition.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

Functions

- KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::RelativePosition &)
- Q_DECLARE_TYPEINFO (KDChart::RelativePosition, Q_MOVABLE_TYPE)

8.80.1 Function Documentation

8.80.1.1 KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::RelativePosition &)

Definition at line 210 of file KDChartRelativePosition.cpp.

References KDChart::RelativePosition::alignment(), KDChart::RelativePosition::horizontalPadding(), KDChart::RelativePosition::referenceArea(), KDChart::RelativePosition::referencePosition(), KDChart::RelativePosition::rotation(), and KDChart::RelativePosition::verticalPadding().

```
211 {
212     dbg << "KDChart::RelativePosition("
213         << "referencearea="<<rp.referenceArea()
214         << "referenceposition="<<rp.referencePosition()
215         << "alignment="<<rp.alignment()
216         << "horizontalpadding="<<rp.horizontalPadding()
217         << "verticalpadding="<<rp.verticalPadding()
218         << "rotation="<<rp.rotation()
219         << ")";
220     return dbg;
221 }
```

8.80.1.2 Q_DECLARE_TYPEINFO (KDChart::RelativePosition, Q_MOVABLE_TYPE)

8.81 KDChartRingDiagram.cpp File Reference

```
#include <QPainter>
#include "KDChartAttributesModel.h"
#include "KDChartPaintContext.h"
#include "KDChartRingDiagram.h"
#include "KDChartRingDiagram_p.h"
#include "KDChartPainterSaver_p.h"
#include "KDChartPieAttributes.h"
#include "KDChartThreeDPieAttributes.h"
#include "KDChartDataValueAttributes.h"
#include <KDABLibFakes>
```

Include dependency graph for KDChartRingDiagram.cpp:

Defines

- #define d d_func()

8.81.1 Define Documentation

8.81.1.1 #define d d_func()

Definition at line 48 of file KDChartRingDiagram.cpp.

8.82 KDChartRingDiagram.h File Reference

`#include "KDChartAbstractPieDiagram.h"`

Include dependency graph for KDChartRingDiagram.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.83 KDChartSignalCompressor.cpp File Reference

```
#include "KDChartSignalCompressor.h"
```

Include dependency graph for KDChartSignalCompressor.cpp:

8.84 KDChartSignalCompressor.h File Reference

`#include <QObject>`

`#include <QTimer>`

Include dependency graph for KDChartSignalCompressor.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.85 KDChartTextArea.cpp File Reference

```
#include "KDChartTextArea.h"
#include "KDChartTextArea_p.h"
#include <qglobal.h>
#include <QPainter>
#include <QRect>
#include <KDABLibFakes>
```

Include dependency graph for KDChartTextArea.cpp:

8.86 KDChartTextArea.h File Reference

`#include <QObject>`

`#include "KDChartGlobal.h"`

`#include "KDChartAbstractAreaBase.h"`

`#include "KDChartLayoutItems.h"`

Include dependency graph for KDChartTextArea.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.87 KDChartTextAttributes.cpp File Reference

`#include "KDChartTextAttributes.h"`

`#include <QFont>`

`#include <QPen>`

`#include <qglobal.h>`

`#include <QApplication>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartTextAttributes.cpp:

Defines

- #define d d_func()

Functions

- QDebug operator<< (QDebug dbg, const KDChart::TextAttributes &ta)

8.87.1 Define Documentation

8.87.1.1 #define d d_func()

Definition at line 34 of file KDChartTextAttributes.cpp.

8.87.2 Function Documentation

8.87.2.1 QDebug operator<< (QDebug *dbg*, const KDChart::TextAttributes & *ta*)

Definition at line 233 of file KDChartTextAttributes.cpp.

```
{
    dbg << "KDChart::TextAttributes("
        << "visible="<<ta.isVisible()
        << "font="<<ta.font().toString() /* What? No QDebug for QFont? */
        << "fontsize="<<ta.fontSize()
        << "minimalfontsize="<<ta.minimalFontSize()
        << "autorotate="<<ta.autoRotate()
        << "autoshrink="<<ta.autoShrink()
        << "rotation="<<ta.rotation()
        << "pen="<<ta.pen()
        << ")";
    return dbg;
}
```

8.88 KDChartTextAttributes.h File Reference

`#include <QDebug>`

`#include <QMetaType>`

`#include "KDChartGlobal.h"`

`#include "KDChartMeasure.h"`

Include dependency graph for KDChartTextAttributes.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

Functions

- KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::TextAttributes &)
- Q_DECLARE_TYPEINFO (KDChart::TextAttributes, Q_MOVABLE_TYPE)

8.88.1 Function Documentation

8.88.1.1 KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::TextAttributes &)

Definition at line 233 of file KDChartTextAttributes.cpp.

References KDChart::TextAttributes::autoRotate(), KDChart::TextAttributes::autoShrink(), KDChart::TextAttributes::font(), KDChart::TextAttributes::fontSize(), KDChart::TextAttributes::isVisible(), KDChart::TextAttributes::minimalFontSize(), KDChart::TextAttributes::pen(), and KDChart::TextAttributes::rotation().

```
234 {
235     dbg << "KDChart::TextAttributes("
236         << "visible="<<ta.isVisible()
237         << "font="<<ta.font().toString() /* What? No QDebug for QFont? */
238         << "fontsize="<<ta.fontSize()
239         << "minimalfontsize="<<ta.minimalFontSize()
240         << "autorotate="<<ta.autoRotate()
241         << "autoshrink="<<ta.autoShrink()
242         << "rotation="<<ta.rotation()
243         << "pen="<<ta.pen()
244         << ")";
245     return dbg;
246 }
```

8.88.1.2 Q_DECLARE_TYPEINFO (KDChart::TextAttributes, Q_MOVABLE_TYPE)

8.89 KDChartTextLabelCache.cpp File Reference

`#include <cmath>`

`#include <QtDebug>`

`#include <QImage>`

`#include <QPixmap>`

`#include <QPainter>`

`#include <QApplication>`

`#include "KDChartTextLabelCache.h"`

Include dependency graph for KDChartTextLabelCache.cpp:

Defines

- #define DUMP_CACHE_STATS
- #define INC_HIT_COUNT { ++HitCount; }
- #define INC_MISS_COUNT { ++MissCount; }

Variables

- int HitCount = 0
- int MissCount = 0

8.89.1 Define Documentation

8.89.1.1 #define DUMP_CACHE_STATS

Value:

```
if ( HitCount != 0 && MissCount != 0 ) { \
        int total = HitCount + MissCount; \
        double hitQuote = ( 1.0 * HitCount ) / total; \
        qDebug() << "PrerenderedLabel dtor: hits/misses/total:" \
        << HitCount << "/" << MissCount << "/" << total \
                 << "(" << 100 * hitQuote << "% hits)"; \
    }
```

Definition at line 16 of file KDChartTextLabelCache.cpp.

Referenced by PrerenderedLabel::~PrerenderedLabel().

8.89.1.2 #define INC_HIT_COUNT { ++HitCount; }

Definition at line 14 of file KDChartTextLabelCache.cpp.

Referenced by PrerenderedLabel::pixmap(), and PrerenderedLabel::referencePointLocation().

8.89.1.3 #define INC_MISS_COUNT { ++MissCount; }

Definition at line 15 of file KDChartTextLabelCache.cpp.

Referenced by PrerenderedLabel::pixmap(), and PrerenderedLabel::referencePointLocation().

8.89.2 Variable Documentation

8.89.2.1 int HitCount = 0

Definition at line 12 of file KDChartTextLabelCache.cpp.

8.89.2.2 int MissCount = 0

Definition at line 13 of file KDChartTextLabelCache.cpp.

8.90 KDChartTextLabelCache.h File Reference

`#include <QPixmap>`

`#include "KDChartEnums.h"`

Include dependency graph for KDChartTextLabelCache.h:

This graph shows which files directly or indirectly include this file:

Classes

- class PrerenderedElement
- class PrerenderedLabel

  CachedLabel is an internal KDChart class that simplifies creation and caching of cached text labels.

8.91 KDChartThreeDBarAttributes.cpp File Reference

`#include "KDChartThreeDBarAttributes.h"`

`#include "KDChartThreeDBarAttributes_p.h"`

`#include <QDebug>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartThreeDBarAttributes.cpp:

Defines

- #define d d_func()

Functions

- QDebug operator<< (QDebug dbg, const KDChart::ThreeDBarAttributes &a)

8.91.1 Define Documentation

8.91.1.1 #define d d_func()

Definition at line 33 of file KDChartThreeDBarAttributes.cpp.

8.91.2 Function Documentation

8.91.2.1 QDebug operator<< (QDebug *dbg*, const KDChart::ThreeDBarAttributes & *a*)

Definition at line 105 of file KDChartThreeDBarAttributes.cpp.

```
{
    dbg << "KDChart::ThreeDBarAttributes(";
    dbg = operator <<( dbg, static_cast<const AbstractThreeDAttributes&>(a) );
    dbg << "useShadowColors="<< a.useShadowColors()
        << "angle=" << a.angle() << ")";
    return dbg;
}
```

8.92 KDChartThreeDBarAttributes.h File Reference

`#include <QMetaType>`

`#include "KDChartAbstractThreeDAttributes.h"`

`#include "KDChartGlobal.h"`

Include dependency graph for KDChartThreeDBarAttributes.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

Functions

- KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::ThreeDBarAttributes &)
- Q_DECLARE_TYPEINFO (KDChart::ThreeDBarAttributes, Q_MOVABLE_TYPE)

8.92.1 Function Documentation

8.92.1.1 KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::ThreeDBarAttributes &)

Definition at line 105 of file KDChartThreeDBarAttributes.cpp.

References KDChart::ThreeDBarAttributes::angle(), and KDChart::ThreeDBarAttributes::useShadowColors().

```
{
    dbg << "KDChart::ThreeDBarAttributes(";
    dbg = operator <<( dbg, static_cast<const AbstractThreeDAttributes&>(a) );
    dbg << "useShadowColors="<< a.useShadowColors()
        << "angle=" << a.angle() << ")";
    return dbg;
}
```

8.92.1.2 Q_DECLARE_TYPEINFO (KDChart::ThreeDBarAttributes, Q_MOVABLE_TYPE)

8.93 KDChartThreeDLineAttributes.cpp File Reference

`#include "KDChartThreeDLineAttributes.h"`

`#include "KDChartThreeDLineAttributes_p.h"`

`#include <QDebug>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartThreeDLineAttributes.cpp:

Defines

- #define d d_func()

Functions

- QDebug operator<< (QDebug dbg, const KDChart::ThreeDLineAttributes &a)

8.93.1 Define Documentation

8.93.1.1 #define d d_func()

Definition at line 33 of file KDChartThreeDLineAttributes.cpp.

8.93.2 Function Documentation

8.93.2.1 QDebug operator<< (QDebug *dbg*, const KDChart::ThreeDLineAttributes & *a*)

Definition at line 106 of file KDChartThreeDLineAttributes.cpp.

```
{
    dbg << "KDChart::ThreeDLineAttributes(";
    dbg = operator <<( dbg, static_cast<const AbstractThreeDAttributes&>(a) );
    dbg << " lineXRotation="<< a.lineXRotation()
        << " lineYRotation="<< a.lineYRotation()
        << ")";
    return dbg;
}
```

8.94 KDChartThreeDLineAttributes.h File Reference

`#include <QMetaType>`

`#include "KDChartAbstractThreeDAttributes.h"`

`#include "KDChartGlobal.h"`

Include dependency graph for KDChartThreeDLineAttributes.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

Functions

- KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::ThreeDLineAttributes &)
- Q_DECLARE_TYPEINFO (KDChart::ThreeDLineAttributes, Q_MOVABLE_TYPE)

8.94.1 Function Documentation

8.94.1.1 KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::ThreeDLineAttributes &)

Definition at line 106 of file KDChartThreeDLineAttributes.cpp.

References KDChart::ThreeDLineAttributes::lineXRotation(), and KDChart::ThreeDLineAttributes::lineYRotation().

```
{
    dbg << "KDChart::ThreeDLineAttributes(";
    dbg = operator <<( dbg, static_cast<const AbstractThreeDAttributes&>(a) );
    dbg << " lineXRotation="<< a.lineXRotation()
        << " lineYRotation="<< a.lineYRotation()
        << ")";
    return dbg;
}
```

8.94.1.2 Q_DECLARE_TYPEINFO (KDChart::ThreeDLineAttributes, Q_MOVABLE_TYPE)

8.95 KDChartThreeDPieAttributes.cpp File Reference

`#include "KDChartThreeDPieAttributes.h"`

`#include "KDChartThreeDPieAttributes_p.h"`

`#include <QDebug>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartThreeDPieAttributes.cpp:

Defines

- #define d d_func()

Functions

- QDebug operator<< (QDebug dbg, const KDChart::ThreeDPieAttributes &a)

8.95.1 Define Documentation

8.95.1.1 #define d d_func()

Definition at line 33 of file KDChartThreeDPieAttributes.cpp.

8.95.2 Function Documentation

8.95.2.1 QDebug operator<< (QDebug *dbg*, const KDChart::ThreeDPieAttributes & *a*)

Definition at line 92 of file KDChartThreeDPieAttributes.cpp.

References KDChart::ThreeDPieAttributes::useShadowColors().

```
{
    dbg << "KDChart::ThreeDPieAttributes(";
    dbg = operator <<( dbg, static_cast<const AbstractThreeDAttributes&>(a) );
    dbg << "useShadowColors="<< a.useShadowColors() << ")";
    return dbg;
}
```

8.96 KDChartThreeDPieAttributes.h File Reference

`#include <QMetaType>`

`#include "KDChartAbstractThreeDAttributes.h"`

`#include "KDChartGlobal.h"`

Include dependency graph for KDChartThreeDPieAttributes.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

Functions

- KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::ThreeDPieAttributes &)
- Q_DECLARE_TYPEINFO (KDChart::ThreeDPieAttributes, Q_MOVABLE_TYPE)

8.96.1 Function Documentation

8.96.1.1 KDCHART_EXPORT QDebug operator<< (QDebug, const KDChart::ThreeDPieAttributes &)

Definition at line 92 of file KDChartThreeDPieAttributes.cpp.

References KDChart::ThreeDPieAttributes::useShadowColors().

```
{
    dbg << "KDChart::ThreeDPieAttributes(";
    dbg = operator <<( dbg, static_cast<const AbstractThreeDAttributes&>(a) );
    dbg << "useShadowColors="<< a.useShadowColors() << ")";
    return dbg;
}
```

8.96.1.2 Q_DECLARE_TYPEINFO (KDChart::ThreeDPieAttributes, Q_MOVABLE_TYPE)

8.97 KDChartWidget.cpp File Reference

`#include <KDChartWidget.h>`

`#include <KDChartWidget_p.h>`

`#include <KDChartAbstractDiagram.h>`

`#include <KDChartBarDiagram.h>`

`#include <KDChartCartesianCoordinatePlane.h>`

`#include <KDChartChart.h>`

`#include <KDChartAbstractCoordinatePlane.h>`

`#include <KDChartLineDiagram.h>`

`#include <KDChartPieDiagram.h>`

`#include <KDChartPolarCoordinatePlane.h>`

`#include <KDChartPolarDiagram.h>`

`#include <KDChartRingDiagram.h>`

`#include <KDChartLegend.h>`

`#include <QDebug>`

`#include <KDABLibFakes>`

Include dependency graph for KDChartWidget.cpp:

Defines

- #define d d_func()
- #define SET_SUB_TYPE(DIAGRAM, SUBTYPE)
- #define TEST_SUB_TYPE(DIAGRAM, INTERNALSUBTYPE, SUBTYPE)

Functions

- bool isCartesian (KDChart::Widget::ChartType type)
- bool isPolar (KDChart::Widget::ChartType type)

8.97.1 Define Documentation

8.97.1.1 #define d d_func()

Definition at line 49 of file KDChartWidget.cpp.

8.97.1.2 #define SET_SUB_TYPE(DIAGRAM, SUBTYPE)

Value:

```
{ \
    if( DIAGRAM ) \
        DIAGRAM->setType( SUBTYPE ); \
}
```

Referenced by KDChart::Widget::setSubType().

8.97.1.3 #define TEST_SUB_TYPE(DIAGRAM, INTERNALSUBTYPE, SUBTYPE)

Value:

```
{ \
    if( DIAGRAM && DIAGRAM->type() == INTERNALSUBTYPE ) \
        retVal = SUBTYPE; \
}
```

8.97.2 Function Documentation

8.97.2.1 bool isCartesian (KDChart::Widget::ChartType *type*) `[static]`

Definition at line 385 of file KDChartWidget.cpp.

```
{
    return (type == KDChart::Widget::Bar || type == KDChart::Widget::Line);
}
```

8.97.2.2 bool isPolar (KDChart::Widget::ChartType *type*) `[static]`

Definition at line 390 of file KDChartWidget.cpp.

```
{
    return (type == KDChart::Widget::Pie
            || type == KDChart::Widget::Ring
            || type == KDChart::Widget::Polar );
}
```

8.98 KDChartWidget.h File Reference

`#include "KDChartGlobal.h"`

`#include <QWidget>`

`#include "KDChartEnums.h"`

`#include "KDChartHeaderFooter.h"`

Include dependency graph for KDChartWidget.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace KDChart

8.99 KDChartZoomParameters.h File Reference

Namespaces

- namespace KDChart

8.100 KDTextDocument.cpp File Reference

```
#include "KDTextDocument.h"
#include <QRect>
#include <QAbstractTextDocumentLayout>
#include <QtDebug>
#include <QTextBlock>
#include <KDABLibFakes>
```

Include dependency graph for KDTextDocument.cpp:

8.101 KDTextDocument.h File Reference

`#include <QTextDocument>`

`#include <QSize>`

Include dependency graph for KDTextDocument.h:

This graph shows which files directly or indirectly include this file:

Classes

- class KDTextDocument

Chapter 9

KD Chart 2 Page Documentation

9.1 Deprecated List

Member KDChart::AbstractPieDiagram::setStartPosition(int degrees) Use PolarCoordinatePlane::setStartPosition(qreal degrees) instead.

Member KDChart::AbstractPieDiagram::startPosition() const Use qreal PolarCoordinatePlane::startPosition instead.

Member KDChart::PolarDiagram::setZeroDegreePosition(int degrees) Use PolarCoordinatePlane::setStartPosition(qreal degrees) instead.

Member KDChart::PolarDiagram::zeroDegreePosition() const Use qreal PolarCoordinatePlane::startPosition instead.

Index

∼AbstractArea
    KDChart::AbstractArea, 35
∼AbstractAreaBase
    KDChart::AbstractAreaBase, 46
∼AbstractAreaWidget
    KDChart::AbstractAreaWidget, 52
∼AbstractAxis
    KDChart::AbstractAxis, 63
∼AbstractCartesianDiagram
    KDChart::AbstractCartesianDiagram, 84
∼AbstractCoordinatePlane
    KDChart::AbstractCoordinatePlane, 119
∼AbstractDiagram
    KDChart::AbstractDiagram, 145
∼AbstractPieDiagram
    KDChart::AbstractPieDiagram, 182
∼AbstractPolarDiagram
    KDChart::AbstractPolarDiagram, 218
∼AbstractThreeDAttributes
    KDChart::AbstractThreeDAttributes, 249
∼AttributesModel
    KDChart::AttributesModel, 254
∼BackgroundAttributes
    KDChart::BackgroundAttributes, 275
∼BarAttributes
    KDChart::BarAttributes, 277
∼BarDiagram
    KDChart::BarDiagram, 286
∼CartesianAxis
    KDChart::CartesianAxis, 330
∼CartesianCoordinatePlane
    KDChart::CartesianCoordinatePlane, 368
∼Chart
    KDChart::Chart, 410
∼DataValueAttributes
    KDChart::DataValueAttributes, 442
∼DiagramObserver
    KDChart::DiagramObserver, 449
∼FrameAttributes
    KDChart::FrameAttributes, 452
∼GlobalMeasureScaling
    KDChart::GlobalMeasureScaling, 454
∼GridAttributes
    KDChart::GridAttributes, 458
∼HeaderFooter
    KDChart::HeaderFooter, 462
∼KDTextDocument
    KDTextDocument, 494
∼Legend
    KDChart::Legend, 500
∼LineAttributes
    KDChart::LineAttributes, 534
∼LineDiagram
    KDChart::LineDiagram, 542
∼MarkerAttributes
    KDChart::MarkerAttributes, 600
∼PaintContext
    KDChart::PaintContext, 615
∼Palette
    KDChart::Palette, 618
∼PieAttributes
    KDChart::PieAttributes, 620
∼PieDiagram
    KDChart::PieDiagram, 628
∼PolarCoordinatePlane
    KDChart::PolarCoordinatePlane, 669
∼PolarDiagram
    KDChart::PolarDiagram, 703
∼PrerenderedElement
    PrerenderedElement, 749
∼PrerenderedLabel
    PrerenderedLabel, 753
∼RelativePosition
    KDChart::RelativePosition, 767
∼RingDiagram
    KDChart::RingDiagram, 776
∼TextArea
    KDChart::TextArea, 815
∼TextAttributes
    KDChart::TextAttributes, 830
∼ThreeDBarAttributes
    KDChart::ThreeDBarAttributes, 844
∼ThreeDLineAttributes
    KDChart::ThreeDLineAttributes, 849
∼ThreeDPieAttributes
    KDChart::ThreeDPieAttributes, 854
∼Widget
    KDChart::Widget, 866
__kdab__dereference_for_methodcall
    KDChartGlobal.h, 940

__pad0__
  KDChart::AbstractArea, 44
  KDChart::AbstractAreaWidget, 60
  KDChart::AbstractAxis, 78
  KDChart::AbstractCartesianDiagram, 114
  KDChart::AbstractCoordinatePlane, 140
  KDChart::AbstractDiagram, 172
  KDChart::AbstractPieDiagram, 213
  KDChart::AbstractPolarDiagram, 246
  KDChart::AttributesModel, 267
  KDChart::BarDiagram, 325
  KDChart::CartesianAxis, 362
  KDChart::CartesianCoordinatePlane, 406
  KDChart::Chart, 426
  KDChart::DatasetSelectorWidget, 439
  KDChart::DiagramObserver, 451
  KDChart::HeaderFooter, 477
  KDChart::Legend, 532
  KDChart::LineDiagram, 586
  KDChart::Palette, 619
  KDChart::PieDiagram, 664
  KDChart::PolarCoordinatePlane, 696
  KDChart::PolarDiagram, 735
  KDChart::RingDiagram, 810
  KDChart::SignalCompressor, 811
  KDChart::TextArea, 828

AbstractArea
  KDChart::AbstractArea, 35
AbstractAreaBase
  KDChart::AbstractAreaBase, 46
AbstractAreaWidget
  KDChart::AbstractAreaWidget, 52
AbstractAxis
  KDChart::AbstractAxis, 63
AbstractCartesianDiagram
  KDChart::AbstractCartesianDiagram, 84
AbstractCoordinatePlane
  KDChart::AbstractCoordinatePlane, 119
AbstractDiagram
  KDChart::AbstractDiagram, 145
AbstractDiagramList
  KDChart, 22
AbstractLayoutItem
  KDChart::AbstractLayoutItem, 173
AbstractPieDiagram
  KDChart::AbstractPieDiagram, 182
AbstractPolarDiagram
  KDChart::AbstractPolarDiagram, 218
AbstractProxyModel
  KDChart::AbstractProxyModel, 247
AbstractThreeDAttributes
  KDChart::AbstractThreeDAttributes, 249
activateTheLayout
  KDChart::Legend, 501
ADD_AUTO_SPACER_IF_NEEDED
  KDChartChart.cpp, 920
ADD_VBOX_WITH_LEGENDS
  KDChartChart.cpp, 920
addAxis
  KDChart::AbstractCartesianDiagram, 84
  KDChart::BarDiagram, 286
  KDChart::LineDiagram, 542
addBrush
  KDChart::Palette, 618
addCoordinatePlane
  KDChart::Chart, 411
addDiagram
  KDChart::AbstractCoordinatePlane, 119
  KDChart::CartesianCoordinatePlane, 368
  KDChart::Legend, 501
  KDChart::PolarCoordinatePlane, 670
addHeaderFooter
  KDChart::Chart, 411
  KDChart::Widget, 867
addLegend
  KDChart::Chart, 412
  KDChart::Widget, 867, 868
adjustedToMaxEmptyInnerPercentage
  KDChart::CartesianCoordinatePlane, 369
adjustHorizontalRangeToData
  KDChart::CartesianCoordinatePlane, 369
adjustLowerBoundToGrid
  KDChart::GridAttributes, 458
adjustUpperBoundToGrid
  KDChart::GridAttributes, 458
adjustVerticalRangeToData
  KDChart::CartesianCoordinatePlane, 370
alignment
  KDChart::Legend, 501
  KDChart::RelativePosition, 767
alignToReferencePoint
  KDChart::AbstractArea, 35
  KDChart::AbstractAreaBase, 46
  KDChart::AbstractAreaWidget, 52
  KDChart::AbstractAxis, 64
  KDChart::AbstractCoordinatePlane, 120
  KDChart::CartesianAxis, 330
  KDChart::CartesianCoordinatePlane, 370
  KDChart::HeaderFooter, 463
  KDChart::Legend, 502
  KDChart::PolarCoordinatePlane, 670
  KDChart::TextArea, 815
allHeadersFooters
  KDChart::Widget, 868
allLegends
  KDChart::Widget, 868
allowOverlappingDataValueTexts

KDChart::AbstractCartesianDiagram, 84
KDChart::AbstractDiagram, 146
KDChart::AbstractPieDiagram, 182
KDChart::AbstractPolarDiagram, 219
KDChart::BarDiagram, 286
KDChart::LineDiagram, 542
KDChart::PieDiagram, 628
KDChart::PolarDiagram, 703
KDChart::RingDiagram, 776

angle
KDChart::ThreeDBarAttributes, 845
PrerenderedLabel, 753

angleUnit
KDChart::PolarCoordinatePlane, 670

antiAliasing
KDChart::AbstractCartesianDiagram, 85
KDChart::AbstractDiagram, 146
KDChart::AbstractPieDiagram, 182
KDChart::AbstractPolarDiagram, 219
KDChart::BarDiagram, 286
KDChart::LineDiagram, 543
KDChart::PieDiagram, 628
KDChart::PolarDiagram, 703
KDChart::RingDiagram, 776

areaGeometry
KDChart::AbstractArea, 35
KDChart::AbstractAreaBase, 46
KDChart::AbstractAreaWidget, 52
KDChart::AbstractAxis, 64
KDChart::AbstractCoordinatePlane, 120
KDChart::CartesianAxis, 330
KDChart::CartesianCoordinatePlane, 370
KDChart::HeaderFooter, 463
KDChart::Legend, 502
KDChart::PolarCoordinatePlane, 671
KDChart::TextArea, 815

AttributesModel
KDChart::AttributesModel, 254

attributesModel
KDChart::AbstractCartesianDiagram, 85
KDChart::AbstractDiagram, 146
KDChart::AbstractPieDiagram, 182
KDChart::AbstractPolarDiagram, 219
KDChart::BarDiagram, 287
KDChart::LineDiagram, 543
KDChart::PieDiagram, 629
KDChart::PolarDiagram, 703
KDChart::RingDiagram, 777

attributesModelRootIndex
KDChart::AbstractCartesianDiagram, 85
KDChart::AbstractDiagram, 147
KDChart::AbstractPieDiagram, 183
KDChart::AbstractPolarDiagram, 219
KDChart::BarDiagram, 287
KDChart::LineDiagram, 543
KDChart::PieDiagram, 629
KDChart::PolarDiagram, 704
KDChart::RingDiagram, 777

autoAdjustGridToZoom
KDChart::CartesianCoordinatePlane, 371

autoAdjustHorizontalRangeToData
KDChart::CartesianCoordinatePlane, 371

autoAdjustVerticalRangeToData
KDChart::CartesianCoordinatePlane, 371

autoReferenceArea
KDChart::HeaderFooter, 463
KDChart::TextArea, 816
KDChart::TextLayoutItem, 836

autoRotate
KDChart::TextAttributes, 830

autoShrink
KDChart::TextAttributes, 830

AutoSpacerLayoutItem
KDChart::AutoSpacerLayoutItem, 268

axes
KDChart::AbstractCartesianDiagram, 86
KDChart::BarDiagram, 287
KDChart::LineDiagram, 544

AxesCalcMode
KDChart::AbstractCoordinatePlane, 118
KDChart::CartesianCoordinatePlane, 368
KDChart::PolarCoordinatePlane, 669

axesCalcModeX
KDChart::CartesianCoordinatePlane, 372

axesCalcModeY
KDChart::CartesianCoordinatePlane, 372

BackgroundAttributes, 15
KDChart::BackgroundAttributes, 275

backgroundAttributes
KDChart::AbstractArea, 35
KDChart::AbstractAreaBase, 46
KDChart::AbstractAreaWidget, 53
KDChart::AbstractAxis, 64
KDChart::AbstractCoordinatePlane, 120
KDChart::CartesianAxis, 331
KDChart::CartesianCoordinatePlane, 372
KDChart::Chart, 413
KDChart::DataValueAttributes, 442
KDChart::HeaderFooter, 463
KDChart::Legend, 502
KDChart::PolarCoordinatePlane, 671
KDChart::TextArea, 816

BackgroundPixmapMode
KDChart::BackgroundAttributes, 274

BackgroundPixmapModeCentered
KDChart::BackgroundAttributes, 274

BackgroundPixmapModeNone

KDChart::BackgroundAttributes, 274
BackgroundPixmapModeScaled
    KDChart::BackgroundAttributes, 274
BackgroundPixmapModeStretched
    KDChart::BackgroundAttributes, 274
Bar
    KDChart::Widget, 866
BarAttributes, 16
    KDChart::BarAttributes, 277
barAttributes
    KDChart::BarDiagram, 288
BarAttributesRole
    KDChart, 24
BarDiagram
    KDChart::BarDiagram, 285
barDiagram
    KDChart::Widget, 868
barGapFactor
    KDChart::BarAttributes, 277
BarType
    KDChart::BarDiagram, 285
Bottom
    KDChart::CartesianAxis, 329
bottom
    KDChart::Widget, 877
bottomOverlap
    KDChart::AbstractArea, 36
    KDChart::AbstractAxis, 64
    KDChart::AbstractCoordinatePlane, 120
    KDChart::CartesianAxis, 331
    KDChart::CartesianCoordinatePlane, 373
    KDChart::PolarCoordinatePlane, 671
brush
    KDChart::AbstractCartesianDiagram, 86, 87
    KDChart::AbstractDiagram, 147
    KDChart::AbstractPieDiagram, 183, 184
    KDChart::AbstractPolarDiagram, 220
    KDChart::BackgroundAttributes, 275
    KDChart::BarDiagram, 288, 289
    KDChart::Legend, 502
    KDChart::LineDiagram, 544
    KDChart::PieDiagram, 629, 630
    KDChart::PolarDiagram, 704, 705
    KDChart::RingDiagram, 777, 778
    PrerenderedLabel, 753
brushes
    KDChart::Legend, 503
buildLegend
    KDChart::Legend, 503
buildReferenceRect
    KDChartPieDiagram.cpp, 966

calcMode
    KDChart::DataDimension, 429
calculateDataBoundaries
    KDChart::AbstractCartesianDiagram, 87
    KDChart::AbstractDiagram, 148
    KDChart::AbstractPieDiagram, 184
    KDChart::AbstractPolarDiagram, 221
    KDChart::BarDiagram, 289
    KDChart::LineDiagram, 545
    KDChart::PieDiagram, 630
    KDChart::PolarDiagram, 705
    KDChart::RingDiagram, 778
calculatedFont
    KDChart::TextAttributes, 830
calculatedFontSize
    KDChart::TextAttributes, 830
calculatedPoint
    KDChart::RelativePosition, 767
calculatedValue
    KDChart::Measure, 609, 610
calculateNextLabel
    KDChartCartesianAxis.cpp, 914
calculateOverlap
    KDChartCartesianAxis.cpp, 915
calculateRawDataBoundingRect
    KDChart::CartesianCoordinatePlane, 373
calculationMode
    KDChart::Measure, 610
CartesianAxis
    KDChart::CartesianAxis, 330
CartesianAxisList
    KDChart, 22
CartesianCoordinatePlane
    KDChart::CartesianCoordinatePlane, 368
Center
    KDChart::Position, 743
center
    KDChart::ZoomParameters, 879
Chart
    KDChart::Chart, 410
ChartType
    KDChart::Widget, 866
checkInvariants
    KDChart::AbstractCartesianDiagram, 87
    KDChart::AbstractDiagram, 148
    KDChart::AbstractPieDiagram, 184
    KDChart::AbstractPolarDiagram, 221
    KDChart::BarDiagram, 291
    KDChart::LineDiagram, 546
    KDChart::PieDiagram, 631
    KDChart::PolarDiagram, 705
    KDChart::RingDiagram, 779
clone
    KDChart::BarDiagram, 291
    KDChart::HeaderFooter, 463

KDChart::Legend, 507
KDChart::LineDiagram, 547
KDChart::PieDiagram, 631
KDChart::PolarDiagram, 706
KDChart::RingDiagram, 779

closeDatasets
KDChart::PolarDiagram, 706

columnConfig
KDChart::DatasetSelectorWidget, 439

columnCount
KDChart::AbstractPieDiagram, 185
KDChart::AbstractPolarDiagram, 221
KDChart::AttributesModel, 254
KDChart::PieDiagram, 632
KDChart::PolarDiagram, 706
KDChart::RingDiagram, 779

columnToIndex
KDChart::AbstractCartesianDiagram, 87
KDChart::AbstractDiagram, 148
KDChart::AbstractPieDiagram, 185
KDChart::AbstractPolarDiagram, 222
KDChart::BarDiagram, 292
KDChart::LineDiagram, 547
KDChart::PieDiagram, 632
KDChart::PolarDiagram, 707
KDChart::RingDiagram, 780

compare
KDChart::AbstractArea, 36
KDChart::AbstractAreaBase, 46
KDChart::AbstractAreaWidget, 53
KDChart::AbstractAxis, 65
KDChart::AbstractCartesianDiagram, 88, 89
KDChart::AbstractCoordinatePlane, 121
KDChart::AbstractDiagram, 148
KDChart::AbstractPieDiagram, 185
KDChart::AbstractPolarDiagram, 222
KDChart::AttributesModel, 254
KDChart::BarDiagram, 292, 293
KDChart::CartesianAxis, 331, 332
KDChart::CartesianCoordinatePlane, 374
KDChart::HeaderFooter, 464
KDChart::Legend, 507, 508
KDChart::LineDiagram, 547, 549
KDChart::PieDiagram, 632
KDChart::PolarCoordinatePlane, 672
KDChart::PolarDiagram, 707
KDChart::RingDiagram, 780
KDChart::TextArea, 816

compareAttributes
KDChart::AttributesModel, 257

connectSignals
KDChart::AbstractAxis, 65
KDChart::CartesianAxis, 333

ConstAbstractDiagramList
KDChart, 23

ConstDiagramList
KDChart, 23

constDiagrams
KDChart::Legend, 508

coordinatePlane
KDChart::AbstractAxis, 66
KDChart::AbstractCartesianDiagram, 90
KDChart::AbstractDiagram, 150
KDChart::AbstractPieDiagram, 187
KDChart::AbstractPolarDiagram, 223
KDChart::BarDiagram, 294
KDChart::CartesianAxis, 333
KDChart::Chart, 413
KDChart::LineDiagram, 550
KDChart::PaintContext, 615
KDChart::PieDiagram, 634
KDChart::PolarDiagram, 708
KDChart::RingDiagram, 781
KDChart::Widget, 868

coordinatePlaneLayout
KDChart::Chart, 413

CoordinatePlaneList
KDChart, 23

coordinatePlanes
KDChart::Chart, 413

CoordinateTransformationList
KDChart::PolarCoordinatePlane, 669

createObserver
KDChart::AbstractAxis, 66
KDChart::CartesianAxis, 333

currentFactors
KDChart::GlobalMeasureScaling, 455

customizedLabel
KDChart::AbstractAxis, 66
KDChart::CartesianAxis, 334

d
KDChartAbstractArea.cpp, 881
KDChartAbstractAreaBase.cpp, 886
KDChartAbstractAreaWidget.cpp, 888
KDChartAbstractAxis.cpp, 890
KDChartAbstractCartesianDiagram.cpp, 892
KDChartAbstractCoordinatePlane.cpp, 894
KDChartAbstractDiagram.cpp, 896
KDChartAbstractPieDiagram.cpp, 898
KDChartAbstractPolarDiagram.cpp, 900
KDChartAbstractThreeDAttributes.cpp, 904
KDChartBackgroundAttributes.cpp, 908
KDChartBarAttributes.cpp, 910
KDChartBarDiagram.cpp, 912

    KDChartCartesianAxis.cpp, 914
    KDChartCartesianCoordinatePlane.cpp, 917
    KDChartChart.cpp, 920
    KDChartDataValueAttributes.cpp, 927
    KDChartFrameAttributes.cpp, 934
    KDChartGridAttributes.cpp, 941
    KDChartHeaderFooter.cpp, 943
    KDChartLegend.cpp, 949
    KDChartLineAttributes.cpp, 951
    KDChartLineDiagram.cpp, 953
    KDChartMarkerAttributes.cpp, 955
    KDChartPaintContext.cpp, 959
    KDChartPalette.cpp, 961
    KDChartPieAttributes.cpp, 964
    KDChartPieDiagram.cpp, 966
    KDChartPolarCoordinatePlane.cpp, 969
    KDChartPolarDiagram.cpp, 971
    KDChartRelativePosition.cpp, 977
    KDChartRingDiagram.cpp, 979
    KDChartTextAttributes.cpp, 985
    KDChartThreeDBarAttributes.cpp, 990
    KDChartThreeDLineAttributes.cpp, 992
    KDChartThreeDPieAttributes.cpp, 994
    KDChartWidget.cpp, 996
data
    KDChart::AttributesModel, 258, 259
    KDChart::DatasetProxyModel, 431
dataBoundaries
    KDChart::AbstractCartesianDiagram, 90
    KDChart::AbstractDiagram, 150
    KDChart::AbstractPieDiagram, 187
    KDChart::AbstractPolarDiagram, 224
    KDChart::BarDiagram, 294
    KDChart::LineDiagram, 550
    KDChart::PieDiagram, 634
    KDChart::PolarDiagram, 709
    KDChart::RingDiagram, 782
dataChanged
    KDChart::AbstractCartesianDiagram, 90
    KDChart::AbstractDiagram, 151
    KDChart::AbstractPieDiagram, 187
    KDChart::AbstractPolarDiagram, 224
    KDChart::BarDiagram, 295
    KDChart::LineDiagram, 550
    KDChart::PieDiagram, 634
    KDChart::PolarDiagram, 709
    KDChart::RingDiagram, 782
DataDimension
    KDChart::DataDimension, 427, 428
DataDimensionsList
    KDChart, 23
dataHidden
    KDChart::AbstractCartesianDiagram, 91
    KDChart::AbstractDiagram, 151
    KDChart::AbstractPieDiagram, 188
    KDChart::AbstractPolarDiagram, 224
    KDChart::BarDiagram, 295
    KDChart::LineDiagram, 551
    KDChart::PieDiagram, 635
    KDChart::PolarDiagram, 710
    KDChart::RingDiagram, 783
DataHiddenRole
    KDChart, 24
dataLabel
    KDChart::DataValueAttributes, 442
dataMap
    KDChart::AttributesModel, 259
datasetBrushes
    KDChart::AbstractCartesianDiagram, 91
    KDChart::AbstractDiagram, 151
    KDChart::AbstractPieDiagram, 188
    KDChart::AbstractPolarDiagram, 225
    KDChart::BarDiagram, 295
    KDChart::LineDiagram, 551
    KDChart::PieDiagram, 635
    KDChart::PolarDiagram, 710
    KDChart::RingDiagram, 783
DatasetBrushRole
    KDChart, 24
datasetCount
    KDChart::Legend, 509
DatasetDescriptionVector
    KDChart, 23
datasetDimension
    KDChart::AbstractCartesianDiagram, 91
    KDChart::AbstractDiagram, 152
    KDChart::AbstractPieDiagram, 188
    KDChart::AbstractPolarDiagram, 225
    KDChart::BarDiagram, 295
    KDChart::LineDiagram, 551
    KDChart::PieDiagram, 635
    KDChart::PolarDiagram, 710
    KDChart::RingDiagram, 783
datasetLabels
    KDChart::AbstractCartesianDiagram, 92
    KDChart::AbstractDiagram, 152
    KDChart::AbstractPieDiagram, 189
    KDChart::AbstractPolarDiagram, 225
    KDChart::BarDiagram, 296
    KDChart::LineDiagram, 552
    KDChart::PieDiagram, 636
    KDChart::PolarDiagram, 711
    KDChart::RingDiagram, 784
datasetMarkers
    KDChart::AbstractCartesianDiagram, 92
    KDChart::AbstractDiagram, 152
    KDChart::AbstractPieDiagram, 189

- KDChart::AbstractPolarDiagram, 226
- KDChart::BarDiagram, 296
- KDChart::LineDiagram, 552
- KDChart::PieDiagram, 636
- KDChart::PolarDiagram, 711
- KDChart::RingDiagram, 784

DatasetPenRole
- KDChart, 24

datasetPens
- KDChart::AbstractCartesianDiagram, 92
- KDChart::AbstractDiagram, 153
- KDChart::AbstractPieDiagram, 190
- KDChart::AbstractPolarDiagram, 226
- KDChart::BarDiagram, 297
- KDChart::LineDiagram, 553
- KDChart::PieDiagram, 637
- KDChart::PolarDiagram, 711
- KDChart::RingDiagram, 784

DatasetProxyModel
- KDChart::DatasetProxyModel, 431

DatasetSelectorWidget
- KDChart::DatasetSelectorWidget, 437

DataValueAttributes, 17
- KDChart::DataValueAttributes, 442

dataValueAttributes
- KDChart::AbstractCartesianDiagram, 93, 94
- KDChart::AbstractDiagram, 153, 154
- KDChart::AbstractPieDiagram, 190, 191
- KDChart::AbstractPolarDiagram, 227
- KDChart::BarDiagram, 297, 298
- KDChart::LineDiagram, 553, 554
- KDChart::PieDiagram, 637, 638
- KDChart::PolarDiagram, 712, 713
- KDChart::RingDiagram, 785, 786

DataValueLabelAttributesRole
- KDChart, 24

decimalDigits
- KDChart::DataValueAttributes, 442

defaultAttributes
- KDChart::DataValueAttributes, 442

defaultAttributesAsVariant
- KDChart::DataValueAttributes, 442

defaultPalette
- KDChart::Palette, 618

deleteObserver
- KDChart::AbstractAxis, 67
- KDChart::CartesianAxis, 334

depth
- KDChart::AbstractThreeDAttributes, 250
- KDChart::ThreeDBarAttributes, 845
- KDChart::ThreeDLineAttributes, 850
- KDChart::ThreeDPieAttributes, 855

diagram
- KDChart::AbstractAxis, 67
- KDChart::AbstractCoordinatePlane, 121
- KDChart::CartesianAxis, 334
- KDChart::CartesianCoordinatePlane, 374
- KDChart::DiagramObserver, 449
- KDChart::Legend, 509
- KDChart::PolarCoordinatePlane, 672
- KDChart::Widget, 869

diagramAttributesChanged
- KDChart::DiagramObserver, 449

diagramDataChanged
- KDChart::DiagramObserver, 449

diagramDataHidden
- KDChart::DiagramObserver, 449

DiagramList
- KDChart, 23

DiagramObserver
- KDChart::DiagramObserver, 448

diagrams
- KDChart::AbstractCoordinatePlane, 121, 122
- KDChart::CartesianCoordinatePlane, 375
- KDChart::Legend, 509
- KDChart::PolarCoordinatePlane, 672, 673

displayArea
- KDChart::LineAttributes, 534

DisplayRoles
- KDChart, 24

distance
- KDChart::DataDimension, 428

doesIsometricScaling
- KDChart::CartesianCoordinatePlane, 375

doItemsLayout
- KDChart::AbstractCartesianDiagram, 94
- KDChart::AbstractDiagram, 154
- KDChart::AbstractPieDiagram, 191
- KDChart::AbstractPolarDiagram, 228
- KDChart::BarDiagram, 298
- KDChart::LineDiagram, 554
- KDChart::PieDiagram, 638
- KDChart::PolarDiagram, 713
- KDChart::RingDiagram, 786

doneSetZoomCenter
- KDChart::CartesianCoordinatePlane, 375

doneSetZoomFactorX
- KDChart::CartesianCoordinatePlane, 376

doneSetZoomFactorY
- KDChart::CartesianCoordinatePlane, 376

drawingArea
- KDChart::CartesianCoordinatePlane, 376

drawSolidExcessArrows
- KDChart::BarAttributes, 277

DUMP_CACHE_STATS
- KDChartTextLabelCache.cpp, 987

East
  KDChart::Position, 743
end
  KDChart::DataDimension, 429
ExcludeCenter
  KDChart::Position, 737
expandingDirections
  KDChart::AbstractCoordinatePlane, 122
  KDChart::AutoSpacerLayoutItem, 269
  KDChart::CartesianAxis, 334
  KDChart::CartesianCoordinatePlane, 377
  KDChart::HeaderFooter, 464
  KDChart::HorizontalLineLayoutItem, 480
  KDChart::LineLayoutItem, 588
  KDChart::LineWithMarkerLayoutItem, 594
  KDChart::MarkerLayoutItem, 603
  KDChart::PolarCoordinatePlane, 673
  KDChart::TextArea, 816
  KDChart::TextLayoutItem, 836
  KDChart::VerticalLineLayoutItem, 859
explode
  KDChart::PieAttributes, 621
explodeFactor
  KDChart::PieAttributes, 621

filterAcceptsColumn
  KDChart::DatasetProxyModel, 431
filterAcceptsRow
  KDChart::DatasetProxyModel, 432
findOrCreateHBoxLayoutByObjectName
  KDChartChart.cpp, 920
findOrCreateLayoutByObjectName
  KDChartChart.cpp, 920
findOrCreateVBoxLayoutByObjectName
  KDChartChart.cpp, 921
firstHeaderFooter
  KDChart::Widget, 869
fixedBarWidth
  KDChart::BarAttributes, 277
fixedDataValueGap
  KDChart::BarAttributes, 277
fixedValueBlockGap
  KDChart::BarAttributes, 278
Floating
  KDChart::Position, 743
floatingPosition
  KDChart::Legend, 510
font
  KDChart::TextAttributes, 831
  PrerenderedLabel, 754
fontSize
  KDChart::TextAttributes, 831
Footer
  KDChart::HeaderFooter, 462
forceRebuild
  KDChart::AbstractAreaWidget, 53
  KDChart::Legend, 510
FrameAttributes, 18
  KDChart::FrameAttributes, 452
frameAttributes
  KDChart::AbstractArea, 36
  KDChart::AbstractAreaBase, 47
  KDChart::AbstractAreaWidget, 53
  KDChart::AbstractAxis, 67
  KDChart::AbstractCoordinatePlane, 122
  KDChart::CartesianAxis, 335
  KDChart::CartesianCoordinatePlane, 377
  KDChart::Chart, 414
  KDChart::DataValueAttributes, 442
  KDChart::HeaderFooter, 465
  KDChart::Legend, 510
  KDChart::PolarCoordinatePlane, 673
  KDChart::TextArea, 817
fromName
  KDChart::Position, 738

geometry
  KDChart::AbstractAxis, 67
  KDChart::AbstractCoordinatePlane, 123
  KDChart::AutoSpacerLayoutItem, 269
  KDChart::CartesianAxis, 335
  KDChart::CartesianCoordinatePlane, 377
  KDChart::HeaderFooter, 465
  KDChart::HorizontalLineLayoutItem, 480
  KDChart::LineLayoutItem, 588
  KDChart::LineWithMarkerLayoutItem, 594
  KDChart::MarkerLayoutItem, 603
  KDChart::PolarCoordinatePlane, 673
  KDChart::TextArea, 817
  KDChart::TextLayoutItem, 836
  KDChart::VerticalLineLayoutItem, 859
getBrush
  KDChart::Palette, 618
getCellValues
  KDChart::LineDiagram, 554
getDataDimensionsList
  KDChart::AbstractCoordinatePlane, 123
  KDChart::CartesianCoordinatePlane, 377
  KDChart::PolarCoordinatePlane, 674
getFrameLeadings
  KDChart::AbstractArea, 37
  KDChart::AbstractAreaBase, 47
  KDChart::AbstractAreaWidget, 54
  KDChart::AbstractAxis, 68
  KDChart::AbstractCoordinatePlane, 123
  KDChart::CartesianAxis, 335

    KDChart::CartesianCoordinatePlane, 378
    KDChart::HeaderFooter, 465
    KDChart::Legend, 511
    KDChart::PolarCoordinatePlane, 674
    KDChart::TextArea, 817
getRawDataBoundingRectFromDiagrams
    KDChart::CartesianCoordinatePlane, 379
globalGridAttributes
    KDChart::AbstractCoordinatePlane, 123
    KDChart::CartesianCoordinatePlane, 379
    KDChart::PolarCoordinatePlane, 674
globalLeadingBottom
    KDChart::Chart, 414
    KDChart::Widget, 869
globalLeadingLeft
    KDChart::Chart, 414
    KDChart::Widget, 869
globalLeadingRight
    KDChart::Chart, 415
    KDChart::Widget, 870
globalLeadingTop
    KDChart::Chart, 415
    KDChart::Widget, 870
GlobalMeasureScaling
    KDChart::GlobalMeasureScaling, 454
granularity
    KDChart::AbstractPieDiagram, 191
    KDChart::PieDiagram, 638
    KDChart::RingDiagram, 786
GranularitySequence
    KDChartEnums, 486
GranularitySequence_10_20
    KDChartEnums, 486
GranularitySequence_10_50
    KDChartEnums, 486
GranularitySequence_125_25
    KDChartEnums, 487
GranularitySequence_25_50
    KDChartEnums, 487
GranularitySequenceIrregular
    KDChartEnums, 487
granularitySequenceToString
    KDChartEnums, 490
GridAttributes, 19
    KDChart::GridAttributes, 458
gridAttributes
    KDChart::CartesianCoordinatePlane, 380
    KDChart::PolarCoordinatePlane, 675
gridDimensionsList
    KDChart::AbstractCoordinatePlane, 124
    KDChart::CartesianCoordinatePlane, 380
    KDChart::PolarCoordinatePlane, 675
gridGranularitySequence
    KDChart::GridAttributes, 458
gridPen
    KDChart::GridAttributes, 458
gridStepWidth
    KDChart::GridAttributes, 458
gridSubStepWidth
    KDChart::GridAttributes, 458
groupGapFactor
    KDChart::BarAttributes, 278

hasAbsoluteFontSize
    KDChart::TextAttributes, 831
hasDefaultTitleTextAttributes
    KDChart::CartesianAxis, 336
hasOwnGridAttributes
    KDChart::CartesianCoordinatePlane, 381
    KDChart::PolarCoordinatePlane, 676
Header
    KDChart::HeaderFooter, 462
headerData
    KDChart::AttributesModel, 259
    KDChart::DatasetProxyModel, 432
HeaderFooter
    KDChart::HeaderFooter, 462
headerFooter
    KDChart::Chart, 415
HeaderFooterList
    KDChart, 23
headerFooters
    KDChart::Chart, 416
HeaderFooterType
    KDChart::HeaderFooter, 462
HitCount
    KDChartTextLabelCache.cpp, 988
horizontalHeaderDataMap
    KDChart::AttributesModel, 260
HorizontalLineLayoutItem
    KDChart::HorizontalLineLayoutItem, 479
horizontalOffset
    KDChart::AbstractCartesianDiagram, 94
    KDChart::AbstractDiagram, 154
    KDChart::AbstractPieDiagram, 192
    KDChart::AbstractPolarDiagram, 228
    KDChart::BarDiagram, 298
    KDChart::LineDiagram, 555
    KDChart::PieDiagram, 639
    KDChart::PolarDiagram, 713
    KDChart::RingDiagram, 786
horizontalPadding
    KDChart::RelativePosition, 767
horizontalRange
    KDChart::CartesianCoordinatePlane, 381

INC_HIT_COUNT
    KDChartTextLabelCache.cpp, 987

INC_MISS_COUNT
  KDChartTextLabelCache.cpp, 987
IncludeCenter
  KDChart::Position, 737
index
  KDChart::AbstractProxyModel, 247
  KDChart::AttributesModel, 260
  KDChart::DatasetProxyModel, 433
indexAt
  KDChart::AbstractCartesianDiagram, 94
  KDChart::AbstractDiagram, 155
  KDChart::AbstractPieDiagram, 192
  KDChart::AbstractPolarDiagram, 228
  KDChart::BarDiagram, 299
  KDChart::LineDiagram, 555
  KDChart::PieDiagram, 639
  KDChart::PolarDiagram, 713
  KDChart::RingDiagram, 787
initFrom
  KDChart::AttributesModel, 261
innerRect
  KDChart::AbstractArea, 37
  KDChart::AbstractAreaBase, 47
  KDChart::AbstractAreaWidget, 54
  KDChart::AbstractAxis, 68
  KDChart::AbstractCoordinatePlane, 124
  KDChart::CartesianAxis, 336
  KDChart::CartesianCoordinatePlane, 382
  KDChart::HeaderFooter, 465
  KDChart::Legend, 511
  KDChart::PolarCoordinatePlane, 677
  KDChart::TextArea, 817
instance
  KDChart::GlobalMeasureScaling, 455
intersects
  KDChart::HeaderFooter, 466, 467
  KDChart::TextArea, 818, 819
  KDChart::TextLayoutItem, 836, 837
invalidate
  PrerenderedElement, 750
  PrerenderedLabel, 754
isAbscissa
  KDChart::CartesianAxis, 336
isCalculated
  KDChart::DataDimension, 429
isCartesian
  KDChartWidget.cpp, 997
isCorner
  KDChart::Position, 739
isEastSide
  KDChart::Position, 739
isEmpty
  KDChart::AbstractCoordinatePlane, 125
  KDChart::AutoSpacerLayoutItem, 269
  KDChart::CartesianAxis, 337
  KDChart::CartesianCoordinatePlane, 382
  KDChart::HeaderFooter, 467
  KDChart::HorizontalLineLayoutItem, 480
  KDChart::LineLayoutItem, 588
  KDChart::LineWithMarkerLayoutItem, 594
  KDChart::MarkerLayoutItem, 603
  KDChart::PolarCoordinatePlane, 677
  KDChart::TextArea, 819
  KDChart::TextLayoutItem, 837
  KDChart::VerticalLineLayoutItem, 859
isEnabled
  KDChart::AbstractThreeDAttributes, 250
  KDChart::ThreeDBarAttributes, 845
  KDChart::ThreeDLineAttributes, 850
  KDChart::ThreeDPieAttributes, 855
isEqualTo
  KDChart::BackgroundAttributes, 275
isFloating
  KDChart::Position, 739
isGridVisible
  KDChart::GridAttributes, 458
isHidden
  KDChart::AbstractCartesianDiagram, 94, 95
  KDChart::AbstractDiagram, 155
  KDChart::AbstractPieDiagram, 192, 193
  KDChart::AbstractPolarDiagram, 228, 229
  KDChart::BarDiagram, 299
  KDChart::LineDiagram, 555, 556
  KDChart::PieDiagram, 639, 640
  KDChart::PolarDiagram, 713, 714
  KDChart::RingDiagram, 787
isIndexHidden
  KDChart::AbstractCartesianDiagram, 95
  KDChart::AbstractDiagram, 156
  KDChart::AbstractPieDiagram, 193
  KDChart::AbstractPolarDiagram, 229
  KDChart::BarDiagram, 300
  KDChart::LineDiagram, 556
  KDChart::PieDiagram, 640
  KDChart::PolarDiagram, 714
  KDChart::RingDiagram, 788
isKnownAttributesRole
  KDChart::AttributesModel, 261
isNorthSide
  KDChart::Position, 739
isNull
  KDChart::PositionPoints, 747
isOrdinate
  KDChart::CartesianAxis, 337
isPolar
  KDChartWidget.cpp, 997

isPole
    KDChart::Position, 740
isSouthSide
    KDChart::Position, 740
isSubGridVisible
    KDChart::GridAttributes, 458
isUnknown
    KDChart::Position, 740
isValid
    KDChart::Palette, 618
isVisible
    KDChart::BackgroundAttributes, 275
    KDChart::DataValueAttributes, 442
    KDChart::FrameAttributes, 452
    KDChart::MarkerAttributes, 600
    KDChart::TextAttributes, 831
isVisiblePoint
    KDChart::AbstractCoordinatePlane, 125
    KDChart::CartesianCoordinatePlane, 382
    KDChart::PolarCoordinatePlane, 677
isWestSide
    KDChart::Position, 740
itemRowLabels
    KDChart::AbstractCartesianDiagram, 95
    KDChart::AbstractDiagram, 156
    KDChart::AbstractPieDiagram, 193
    KDChart::AbstractPolarDiagram, 229
    KDChart::BarDiagram, 300
    KDChart::LineDiagram, 556
    KDChart::PieDiagram, 640
    KDChart::PolarDiagram, 714
    KDChart::RingDiagram, 788

KDAB_SET_OBJECT_NAME
    KDChartGlobal.h, 936
KDChart, 20
    AbstractDiagramList, 22
    BarAttributesRole, 24
    CartesianAxisList, 22
    ConstAbstractDiagramList, 23
    ConstDiagramList, 23
    CoordinatePlaneList, 23
    DataDimensionsList, 23
    DataHiddenRole, 24
    DatasetBrushRole, 24
    DatasetDescriptionVector, 23
    DatasetPenRole, 24
    DataValueLabelAttributesRole, 24
    DiagramList, 23
    DisplayRoles, 24
    HeaderFooterList, 23
    LegendList, 23
    LineAttributesRole, 24
    PieAttributesRole, 24
    ThreeDAttributesRole, 24
    ThreeDBarAttributesRole, 24
    ThreeDLineAttributesRole, 24
    ThreeDPieAttributesRole, 24
KDChart::AbstractArea, 33
KDChart::AbstractArea
    ~AbstractArea, 35
    __pad0__, 44
    AbstractArea, 35
    alignToReferencePoint, 35
    areaGeometry, 35
    backgroundAttributes, 35
    bottomOverlap, 36
    compare, 36
    frameAttributes, 36
    getFrameLeadings, 37
    innerRect, 37
    leftOverlap, 37
    mParent, 44
    mParentLayout, 44
    paint, 38
    paintAll, 38
    paintBackground, 39
    paintBackgroundAttributes, 39
    paintCtx, 40
    paintFrame, 40
    paintFrameAttributes, 40
    paintIntoRect, 41
    parentLayout, 41
    positionHasChanged, 41
    removeFromParentLayout, 41
    rightOverlap, 42
    setBackgroundAttributes, 42
    setFrameAttributes, 42
    setParentLayout, 43
    setParentWidget, 43
    sizeHintChanged, 43
    topOverlap, 43
KDChart::AbstractAreaBase, 45
KDChart::AbstractAreaBase
    ~AbstractAreaBase, 46
    AbstractAreaBase, 46
    alignToReferencePoint, 46
    areaGeometry, 46
    backgroundAttributes, 46
    compare, 46
    frameAttributes, 47
    getFrameLeadings, 47
    innerRect, 47
    paintBackground, 48
    paintBackgroundAttributes, 48
    paintFrame, 49
    paintFrameAttributes, 49
    positionHasChanged, 50

  setBackgroundAttributes, 50
  setFrameAttributes, 50
KDChart::AbstractAreaWidget, 51
KDChart::AbstractAreaWidget
  ~AbstractAreaWidget, 52
  __pad0__, 60
  AbstractAreaWidget, 52
  alignToReferencePoint, 52
  areaGeometry, 52
  backgroundAttributes, 53
  compare, 53
  forceRebuild, 53
  frameAttributes, 53
  getFrameLeadings, 54
  innerRect, 54
  needSizeHint, 54
  paint, 55
  paintAll, 55
  paintBackground, 56
  paintBackgroundAttributes, 56
  paintEvent, 57
  paintFrame, 57
  paintFrameAttributes, 58
  paintIntoRect, 58
  positionHasChanged, 59
  resizeLayout, 59
  setBackgroundAttributes, 59
  setFrameAttributes, 59
KDChart::AbstractAxis, 61
KDChart::AbstractAxis
  ~AbstractAxis, 63
  __pad0__, 78
  AbstractAxis, 63
  alignToReferencePoint, 64
  areaGeometry, 64
  backgroundAttributes, 64
  bottomOverlap, 64
  compare, 65
  connectSignals, 65
  coordinatePlane, 66
  createObserver, 66
  customizedLabel, 66
  deleteObserver, 67
  diagram, 67
  frameAttributes, 67
  geometry, 67
  getFrameLeadings, 68
  innerRect, 68
  labels, 68
  leftOverlap, 69
  mParent, 78
  mParentLayout, 78
  observedBy, 69
  paint, 69
  paintAll, 69
  paintBackground, 70
  paintBackgroundAttributes, 70
  paintCtx, 71
  paintFrame, 72
  paintFrameAttributes, 72
  paintIntoRect, 72
  parentLayout, 73
  positionHasChanged, 73
  Q_SLOTS, 78
  removeFromParentLayout, 73
  rightOverlap, 73
  setBackgroundAttributes, 74
  setFrameAttributes, 74
  setGeometry, 74
  setLabels, 74
  setParentLayout, 75
  setParentWidget, 75
  setShortLabels, 75
  setTextAttributes, 76
  shortLabels, 76
  sizeHintChanged, 76
  textAttributes, 77
  topOverlap, 77
KDChart::AbstractCartesianDiagram, 79
KDChart::AbstractCartesianDiagram
  ~AbstractCartesianDiagram, 84
  __pad0__, 114
  AbstractCartesianDiagram, 84
  addAxis, 84
  allowOverlappingDataValueTexts, 84
  antiAliasing, 85
  attributesModel, 85
  attributesModelRootIndex, 85
  axes, 86
  brush, 86, 87
  calculateDataBoundaries, 87
  checkInvariants, 87
  columnToIndex, 87
  compare, 88, 89
  coordinatePlane, 90
  dataBoundaries, 90
  dataChanged, 90
  dataHidden, 91
  datasetBrushes, 91
  datasetDimension, 91
  datasetLabels, 92
  datasetMarkers, 92
  datasetPens, 92
  dataValueAttributes, 93, 94
  doItemsLayout, 94
  horizontalOffset, 94
  indexAt, 94
  isHidden, 94, 95

isIndexHidden, 95
itemRowLabels, 95
layoutPlanes, 96
modelsChanged, 96
moveCursor, 96
numberOfAbscissaSegments, 96
numberOfOrdinateSegments, 97
paint, 97
paintDataValueText, 97
paintDataValueTexts, 98
paintMarker, 99
paintMarkers, 101
pen, 101, 102
percentMode, 102
propertiesChanged, 102
referenceDiagram, 102
referenceDiagramOffset, 102
resize, 103
scrollTo, 103
setAllowOverlappingDataValueTexts, 103
setAntiAliasing, 103
setAttributesModel, 104
setAttributesModelRootIndex, 104
setBrush, 105
setCoordinatePlane, 105
setDataBoundariesDirty, 106
setDatasetDimension, 106
setDataValueAttributes, 107
setHidden, 108
setModel, 109
setPen, 109, 110
setPercentMode, 110
setReferenceDiagram, 110
setRootIndex, 110
setSelection, 111
takeAxis, 111
threeDItemDepth, 111
update, 111
useDefaultColors, 112
useRainbowColors, 112
usesExternalAttributesModel, 112
useSubduedColors, 112
valueForCell, 113
verticalOffset, 113
visualRect, 113
visualRegionForSelection, 113
KDChart::AbstractCoordinatePlane, 115
Linear, 118
Logarithmic, 118
KDChart::AbstractCoordinatePlane
~AbstractCoordinatePlane, 119
__pad0__, 140
AbstractCoordinatePlane, 119
addDiagram, 119
alignToReferencePoint, 120
areaGeometry, 120
AxesCalcMode, 118
backgroundAttributes, 120
bottomOverlap, 120
compare, 121
diagram, 121
diagrams, 121, 122
expandingDirections, 122
frameAttributes, 122
geometry, 123
getDataDimensionsList, 123
getFrameLeadings, 123
globalGridAttributes, 123
gridDimensionsList, 124
innerRect, 124
isEmpty, 125
isVisiblePoint, 125
layoutDiagrams, 125
layoutPlanes, 125
leftOverlap, 126
maximumSize, 126
minimumSize, 126
minimumSizeHint, 126
mousePressEvent, 127
mParent, 140
mParentLayout, 140
needLayoutPlanes, 127
needRelayout, 127
needUpdate, 127
paint, 127
paintAll, 127
paintBackground, 128
paintBackgroundAttributes, 128
paintCtx, 129
paintFrame, 130
paintFrameAttributes, 130
paintIntoRect, 130
parent, 131
parentLayout, 131
positionHasChanged, 131
propertiesChanged, 131
Q_SLOTS, 140
referenceCoordinatePlane, 132
relayout, 132
removeFromParentLayout, 132
replaceDiagram, 133
rightOverlap, 133
setBackgroundAttributes, 134
setFrameAttributes, 134
setGeometry, 134
setGlobalGridAttributes, 135
setGridNeedsRecalculate, 135
setParent, 135

  setParentLayout, 136
  setParentWidget, 136
  setReferenceCoordinatePlane, 136
  setZoomCenter, 136
  setZoomFactorX, 137
  setZoomFactorY, 137
  sizeHint, 137
  sizeHintChanged, 137
  sizePolicy, 138
  takeDiagram, 138
  topOverlap, 138
  translate, 139
  zoomCenter, 139
  zoomFactorX, 139
  zoomFactorY, 140
KDChart::AbstractDiagram, 141
KDChart::AbstractDiagram
  ∼AbstractDiagram, 145
  __pad0__, 172
  AbstractDiagram, 145
  allowOverlappingDataValueTexts, 146
  antiAliasing, 146
  attributesModel, 146
  attributesModelRootIndex, 147
  brush, 147
  calculateDataBoundaries, 148
  checkInvariants, 148
  columnToIndex, 148
  compare, 148
  coordinatePlane, 150
  dataBoundaries, 150
  dataChanged, 151
  dataHidden, 151
  datasetBrushes, 151
  datasetDimension, 152
  datasetLabels, 152
  datasetMarkers, 152
  datasetPens, 153
  dataValueAttributes, 153, 154
  doItemsLayout, 154
  horizontalOffset, 154
  indexAt, 155
  isHidden, 155
  isIndexHidden, 156
  itemRowLabels, 156
  modelsChanged, 156
  moveCursor, 156
  paint, 157
  paintDataValueText, 157
  paintDataValueTexts, 158
  paintMarker, 159
  paintMarkers, 161
  pen, 161, 162
  percentMode, 162
  propertiesChanged, 162
  resize, 162
  scrollTo, 162
  setAllowOverlappingDataValueTexts, 163
  setAntiAliasing, 163
  setAttributesModel, 163
  setAttributesModelRootIndex, 164
  setBrush, 164, 165
  setCoordinatePlane, 165
  setDataBoundariesDirty, 165
  setDatasetDimension, 165
  setDataValueAttributes, 166
  setHidden, 167
  setModel, 168
  setPen, 168, 169
  setPercentMode, 169
  setRootIndex, 169
  setSelection, 169
  update, 169
  useDefaultColors, 170
  useRainbowColors, 170
  usesExternalAttributesModel, 170
  useSubduedColors, 170
  valueForCell, 171
  verticalOffset, 171
  visualRect, 171
  visualRegionForSelection, 171
KDChart::AbstractLayoutItem, 173
KDChart::AbstractLayoutItem
  AbstractLayoutItem, 173
  mParent, 175
  mParentLayout, 175
  paint, 173
  paintAll, 174
  paintCtx, 174
  parentLayout, 174
  removeFromParentLayout, 174
  setParentLayout, 174
  setParentWidget, 175
  sizeHintChanged, 175
KDChart::AbstractPieDiagram, 177
KDChart::AbstractPieDiagram
  ∼AbstractPieDiagram, 182
  __pad0__, 213
  AbstractPieDiagram, 182
  allowOverlappingDataValueTexts, 182
  antiAliasing, 182
  attributesModel, 182
  attributesModelRootIndex, 183
  brush, 183, 184
  calculateDataBoundaries, 184
  checkInvariants, 184
  columnCount, 185
  columnToIndex, 185

- compare, 185
- coordinatePlane, 187
- dataBoundaries, 187
- dataChanged, 187
- dataHidden, 188
- datasetBrushes, 188
- datasetDimension, 188
- datasetLabels, 189
- datasetMarkers, 189
- datasetPens, 190
- dataValueAttributes, 190, 191
- doItemsLayout, 191
- granularity, 191
- horizontalOffset, 192
- indexAt, 192
- isHidden, 192, 193
- isIndexHidden, 193
- itemRowLabels, 193
- modelsChanged, 194
- moveCursor, 194
- numberOfGridRings, 194
- numberOfValuesPerDataset, 194
- paint, 194
- paintDataValueText, 194
- paintDataValueTexts, 196
- paintMarker, 196, 197
- paintMarkers, 198
- pen, 199
- percentMode, 199
- pieAttributes, 200
- polarCoordinatePlane, 200
- propertiesChanged, 201
- resize, 201
- scrollTo, 201
- setAllowOverlappingDataValueTexts, 201
- setAntiAliasing, 201
- setAttributesModel, 202
- setAttributesModelRootIndex, 202
- setBrush, 203
- setCoordinatePlane, 203
- setDataBoundariesDirty, 204
- setDatasetDimension, 204
- setDataValueAttributes, 204, 205
- setGranularity, 205
- setHidden, 206
- setModel, 207
- setPen, 207, 208
- setPercentMode, 208
- setPieAttributes, 208
- setRootIndex, 209
- setSelection, 209
- setStartPosition, 209
- setThreeDPieAttributes, 209, 210
- startPosition, 210
- threeDPieAttributes, 210, 211
- update, 211
- useDefaultColors, 211
- useRainbowColors, 211
- usesExternalAttributesModel, 212
- useSubduedColors, 212
- valueForCell, 212
- valueTotals, 212
- verticalOffset, 213
- visualRect, 213
- visualRegionForSelection, 213

KDChart::AbstractPolarDiagram, 214

KDChart::AbstractPolarDiagram
- ~AbstractPolarDiagram, 218
- __pad0__, 246
- AbstractPolarDiagram, 218
- allowOverlappingDataValueTexts, 219
- antiAliasing, 219
- attributesModel, 219
- attributesModelRootIndex, 219
- brush, 220
- calculateDataBoundaries, 221
- checkInvariants, 221
- columnCount, 221
- columnToIndex, 222
- compare, 222
- coordinatePlane, 223
- dataBoundaries, 224
- dataChanged, 224
- dataHidden, 224
- datasetBrushes, 225
- datasetDimension, 225
- datasetLabels, 225
- datasetMarkers, 226
- datasetPens, 226
- dataValueAttributes, 227
- doItemsLayout, 228
- horizontalOffset, 228
- indexAt, 228
- isHidden, 228, 229
- isIndexHidden, 229
- itemRowLabels, 229
- modelsChanged, 230
- moveCursor, 230
- numberOfGridRings, 230
- numberOfValuesPerDataset, 230
- paint, 230
- paintDataValueText, 231
- paintDataValueTexts, 232
- paintMarker, 232, 233
- paintMarkers, 234
- pen, 235
- percentMode, 236
- polarCoordinatePlane, 236

propertiesChanged, 236
resize, 236
scrollTo, 236
setAllowOverlappingDataValueTexts, 237
setAntiAliasing, 237
setAttributesModel, 237
setAttributesModelRootIndex, 238
setBrush, 238, 239
setCoordinatePlane, 239
setDataBoundariesDirty, 239
setDatasetDimension, 239
setDataValueAttributes, 240
setHidden, 241, 242
setModel, 242
setPen, 242, 243
setPercentMode, 243
setRootIndex, 243
setSelection, 244
update, 244
useDefaultColors, 244
useRainbowColors, 244
usesExternalAttributesModel, 244
useSubduedColors, 245
valueForCell, 245
valueTotals, 245
verticalOffset, 246
visualRect, 246
visualRegionForSelection, 246
KDChart::AbstractProxyModel, 247
KDChart::AbstractProxyModel
  AbstractProxyModel, 247
  index, 247
  mapFromSource, 247
  mapToSource, 248
  parent, 248
KDChart::AbstractThreeDAttributes, 249
KDChart::AbstractThreeDAttributes
  ∼AbstractThreeDAttributes, 249
  AbstractThreeDAttributes, 249
  depth, 250
  isEnabled, 250
  operator!=, 250
  operator=, 250
  operator==, 250
  setDepth, 251
  setEnabled, 251
  validDepth, 251
KDChart::AttributesModel, 252
  PaletteTypeDefault, 254
  PaletteTypeRainbow, 254
  PaletteTypeSubdued, 254
KDChart::AttributesModel
  ∼AttributesModel, 254
  __pad0__, 267
  AttributesModel, 254
  columnCount, 254
  compare, 254
  compareAttributes, 257
  data, 258, 259
  dataMap, 259
  headerData, 259
  horizontalHeaderDataMap, 260
  index, 260
  initFrom, 261
  isKnownAttributesRole, 261
  mapFromSource, 262
  mapToSource, 262
  modelData, 262
  modelDataMap, 263
  PaletteType, 254
  paletteType, 263
  parent, 263
  resetData, 263
  resetHeaderData, 263
  rowCount, 264
  setData, 264
  setDataMap, 264
  setHeaderData, 265
  setHorizontalHeaderDataMap, 265
  setModelData, 265
  setModelDataMap, 265
  setPaletteType, 266
  setSourceModel, 266
  setVerticalHeaderDataMap, 266
  verticalHeaderDataMap, 266
KDChart::AutoSpacerLayoutItem, 268
KDChart::AutoSpacerLayoutItem
  AutoSpacerLayoutItem, 268
  expandingDirections, 269
  geometry, 269
  isEmpty, 269
  maximumSize, 269
  minimumSize, 269
  mParent, 273
  mParentLayout, 273
  paint, 269
  paintAll, 270
  paintCtx, 271
  parentLayout, 271
  removeFromParentLayout, 271
  setGeometry, 271
  setParentLayout, 271
  setParentWidget, 272
  sizeHint, 272
  sizeHintChanged, 273
KDChart::BackgroundAttributes, 274
  BackgroundPixmapModeCentered, 274
  BackgroundPixmapModeNone, 274

- BackgroundPixmapModeScaled, 274
- BackgroundPixmapModeStretched, 274

KDChart::BackgroundAttributes
- ~BackgroundAttributes, 275
- BackgroundAttributes, 275
- BackgroundPixmapMode, 274
- brush, 275
- isEqualTo, 275
- isVisible, 275
- operator!=, 275
- operator=, 275
- operator==, 275
- pixmap, 275
- pixmapMode, 275
- setBrush, 275
- setPixmap, 276
- setPixmapMode, 276
- setVisible, 276

KDChart::BarAttributes, 277

KDChart::BarAttributes
- ~BarAttributes, 277
- BarAttributes, 277
- barGapFactor, 277
- drawSolidExcessArrows, 277
- fixedBarWidth, 277
- fixedDataValueGap, 277
- fixedValueBlockGap, 278
- groupGapFactor, 278
- operator!=, 278
- operator=, 278
- operator==, 278
- setBarGapFactor, 278
- setDrawSolidExcessArrows, 278
- setFixedBarWidth, 278
- setFixedDataValueGap, 278
- setFixedValueBlockGap, 278
- setGroupGapFactor, 278
- setUseFixedBarWidth, 278
- setUseFixedDataValueGap, 278
- setUseFixedValueBlockGap, 278
- useFixedBarWidth, 278
- useFixedDataValueGap, 278
- useFixedValueBlockGap, 278

KDChart::BarDiagram, 280
- Normal, 285
- Percent, 285
- Rows, 285
- Stacked, 285

KDChart::BarDiagram
- ~BarDiagram, 286
- __pad0__, 325
- addAxis, 286
- allowOverlappingDataValueTexts, 286
- antiAliasing, 286
- attributesModel, 287
- attributesModelRootIndex, 287
- axes, 287
- barAttributes, 288
- BarDiagram, 285
- BarType, 285
- brush, 288, 289
- calculateDataBoundaries, 289
- checkInvariants, 291
- clone, 291
- columnToIndex, 292
- compare, 292, 293
- coordinatePlane, 294
- dataBoundaries, 294
- dataChanged, 295
- dataHidden, 295
- datasetBrushes, 295
- datasetDimension, 295
- datasetLabels, 296
- datasetMarkers, 296
- datasetPens, 297
- dataValueAttributes, 297, 298
- doItemsLayout, 298
- horizontalOffset, 298
- indexAt, 299
- isHidden, 299
- isIndexHidden, 300
- itemRowLabels, 300
- layoutPlanes, 300
- modelsChanged, 301
- moveCursor, 301
- numberOfAbscissaSegments, 301
- numberOfOrdinateSegments, 301
- paint, 301
- paintDataValueText, 305
- paintDataValueTexts, 306
- paintMarker, 307
- paintMarkers, 309
- pen, 309, 310
- percentMode, 310
- propertiesChanged, 310
- referenceDiagram, 310
- referenceDiagramOffset, 311
- resize, 311
- resizeEvent, 311
- scrollTo, 311
- setAllowOverlappingDataValueTexts, 312
- setAntiAliasing, 312
- setAttributesModel, 312
- setAttributesModelRootIndex, 313
- setBarAttributes, 313, 314
- setBrush, 314
- setCoordinatePlane, 314
- setDataBoundariesDirty, 315

- setDatasetDimension, 315
- setDataValueAttributes, 316
- setHidden, 317
- setModel, 318
- setPen, 318, 319
- setPercentMode, 319
- setReferenceDiagram, 319
- setRootIndex, 319
- setSelection, 320
- setThreeDBarAttributes, 320
- setType, 321
- takeAxis, 321
- threeDBarAttributes, 321, 322
- threeDItemDepth, 322
- type, 323
- update, 323
- useDefaultColors, 323
- useRainbowColors, 323
- usesExternalAttributesModel, 324
- useSubduedColors, 324
- valueForCell, 324
- verticalOffset, 324
- visualRect, 325
- visualRegionForSelection, 325

KDChart::CartesianAxis, 326
- Bottom, 329
- Left, 329
- Right, 329
- Top, 329

KDChart::CartesianAxis
- ∼CartesianAxis, 330
- __pad0__, 362
- alignToReferencePoint, 330
- areaGeometry, 330
- backgroundAttributes, 331
- bottomOverlap, 331
- CartesianAxis, 330
- compare, 331, 332
- connectSignals, 333
- coordinatePlane, 333
- createObserver, 333
- customizedLabel, 334
- deleteObserver, 334
- diagram, 334
- expandingDirections, 334
- frameAttributes, 335
- geometry, 335
- getFrameLeadings, 335
- hasDefaultTitleTextAttributes, 336
- innerRect, 336
- isAbscissa, 336
- isEmpty, 337
- isOrdinate, 337
- labels, 337
- layoutPlanes, 337
- leftOverlap, 338
- maximumSize, 338
- minimumSize, 341
- mParent, 362
- mParentLayout, 362
- observedBy, 341
- paint, 341
- paintAll, 342
- paintBackground, 343
- paintBackgroundAttributes, 343
- paintCtx, 344
- paintFrame, 353
- paintFrameAttributes, 353
- paintIntoRect, 354
- parentLayout, 354
- Position, 329
- position, 354
- positionHasChanged, 354
- Q_SLOTS, 362
- removeFromParentLayout, 355
- resetTitleTextAttributes, 355
- rightOverlap, 355
- setBackgroundAttributes, 356
- setFrameAttributes, 356
- setGeometry, 356
- setLabels, 356
- setParentLayout, 357
- setParentWidget, 357
- setPosition, 357
- setShortLabels, 357
- setTextAttributes, 358
- setTitleText, 358
- setTitleTextAttributes, 358
- shortLabels, 359
- sizeHint, 359
- sizeHintChanged, 359
- textAttributes, 360
- tickLength, 360
- titleText, 360
- titleTextAttributes, 361
- topOverlap, 361

KDChart::CartesianCoordinatePlane, 363
- Linear, 368
- Logarithmic, 368

KDChart::CartesianCoordinatePlane
- ∼CartesianCoordinatePlane, 368
- __pad0__, 406
- addDiagram, 368
- adjustedToMaxEmptyInnerPercentage, 369
- adjustHorizontalRangeToData, 369
- adjustVerticalRangeToData, 370
- alignToReferencePoint, 370
- areaGeometry, 370

autoAdjustGridToZoom, 371
autoAdjustHorizontalRangeToData, 371
autoAdjustVerticalRangeToData, 371
AxesCalcMode, 368
axesCalcModeX, 372
axesCalcModeY, 372
backgroundAttributes, 372
bottomOverlap, 373
calculateRawDataBoundingRect, 373
CartesianCoordinatePlane, 368
compare, 374
diagram, 374
diagrams, 375
doesIsometricScaling, 375
doneSetZoomCenter, 375
doneSetZoomFactorX, 376
doneSetZoomFactorY, 376
drawingArea, 376
expandingDirections, 377
frameAttributes, 377
geometry, 377
getDataDimensionsList, 377
getFrameLeadings, 378
getRawDataBoundingRectFromDiagrams, 379
globalGridAttributes, 379
gridAttributes, 380
gridDimensionsList, 380
hasOwnGridAttributes, 381
horizontalRange, 381
innerRect, 382
isEmpty, 382
isVisiblePoint, 382
layoutDiagrams, 383
layoutPlanes, 384
leftOverlap, 384
maximumSize, 385
minimumSize, 385
minimumSizeHint, 385
mousePressEvent, 386
mParent, 406
mParentLayout, 407
needLayoutPlanes, 386
needRelayout, 386
needUpdate, 386
paint, 386
paintAll, 387
paintBackground, 388
paintBackgroundAttributes, 388
paintCtx, 389
paintEvent, 389
paintFrame, 390
paintFrameAttributes, 390
paintIntoRect, 390
parent, 391
parentLayout, 391
positionHasChanged, 391
propertiesChanged, 391
Q_SLOTS, 407
referenceCoordinatePlane, 392
relayout, 392
removeFromParentLayout, 392
replaceDiagram, 393
resetGridAttributes, 393
rightOverlap, 394
setAutoAdjustGridToZoom, 394
setAutoAdjustHorizontalRangeToData, 395
setAutoAdjustVerticalRangeToData, 395
setAxesCalcModes, 396
setAxesCalcModeX, 396
setAxesCalcModeY, 396
setBackgroundAttributes, 397
setFrameAttributes, 397
setGeometry, 397
setGlobalGridAttributes, 398
setGridAttributes, 398
setGridNeedsRecalculate, 399
setHorizontalRange, 399
setIsometricScaling, 400
setParent, 400
setParentLayout, 400
setParentWidget, 400
setReferenceCoordinatePlane, 401
setVerticalRange, 401
setZoomCenter, 402
setZoomFactorX, 402
setZoomFactorY, 402
sizeHint, 403
sizeHintChanged, 403
sizePolicy, 403
takeDiagram, 404
topOverlap, 404
translate, 404
translateBack, 405
verticalRange, 405
zoomCenter, 405
zoomFactorX, 406
zoomFactorY, 406

KDChart::Chart, 408
- ~Chart, 410
- __pad0__, 426
- addCoordinatePlane, 411
- addHeaderFooter, 411
- addLegend, 412
- backgroundAttributes, 413
- Chart, 410
- coordinatePlane, 413

- coordinatePlaneLayout, 413
- coordinatePlanes, 413
- frameAttributes, 414
- globalLeadingBottom, 414
- globalLeadingLeft, 414
- globalLeadingRight, 415
- globalLeadingTop, 415
- headerFooter, 415
- headerFooters, 416
- legend, 416
- legends, 416
- mousePressEvent, 417
- paint, 417
- paintEvent, 418
- reLayoutFloatingLegends, 419
- replaceCoordinatePlane, 419
- replaceHeaderFooter, 420
- replaceLegend, 420
- resizeEvent, 421
- setBackgroundAttributes, 421
- setCoordinatePlaneLayout, 422
- setFrameAttributes, 422
- setGlobalLeading, 422
- setGlobalLeadingBottom, 423
- setGlobalLeadingLeft, 423
- setGlobalLeadingRight, 424
- setGlobalLeadingTop, 424
- takeCoordinatePlane, 424
- takeHeaderFooter, 425
- takeLegend, 425

KDChart::DataDimension, 427

KDChart::DataDimension
- calcMode, 429
- DataDimension, 427, 428
- distance, 428
- end, 429
- isCalculated, 429
- operator!=, 428
- operator==, 428
- sequence, 429
- start, 429
- stepWidth, 429
- subStepWidth, 429

KDChart::DatasetProxyModel, 430

KDChart::DatasetProxyModel
- data, 431
- DatasetProxyModel, 431
- filterAcceptsColumn, 431
- filterAcceptsRow, 432
- headerData, 432
- index, 433
- mapFromSource, 433
- mapToSource, 433
- parent, 434
- Q_SLOTS, 436
- setDatasetColumnDescriptionVector, 434
- setDatasetDescriptionVectors, 434
- setDatasetRowDescriptionVector, 435
- setSourceModel, 435
- setSourceRootIndex, 435

KDChart::DatasetSelectorWidget, 437

KDChart::DatasetSelectorWidget
- __pad0__, 439
- columnConfig, 439
- DatasetSelectorWidget, 437
- mappingDisabled, 437
- on_cbReverseColumns_stateChanged, 437
- on_cbReverseRows_stateChanged, 438
- on_groupBox_toggled, 438
- on_sbColumnCount_valueChanged, 438
- on_sbRowCount_valueChanged, 438
- on_sbStartRow_valueChanged, 438
- Q_SLOTS, 439
- setSourceColumnCount, 439

KDChart::DataValueAttributes, 440

KDChart::DataValueAttributes
- ∼DataValueAttributes, 442
- backgroundAttributes, 442
- dataLabel, 442
- DataValueAttributes, 442
- decimalDigits, 442
- defaultAttributes, 442
- defaultAttributesAsVariant, 442
- frameAttributes, 442
- isVisible, 442
- markerAttributes, 443
- negativePosition, 443
- operator!=, 443
- operator=, 443
- operator==, 443
- position, 443
- positivePosition, 443
- powerOfTenDivisor, 444
- prefix, 444
- setBackgroundAttributes, 444
- setDataLabel, 444
- setDecimalDigits, 444
- setFrameAttributes, 444
- setMarkerAttributes, 445
- setNegativePosition, 445
- setPositivePosition, 445
- setPowerOfTenDivisor, 445
- setPrefix, 445
- setShowInfinite, 446
- setShowRepetitiveDataLabels, 446
- setSuffix, 446
- setTextAttributes, 446
- setVisible, 446

- showInfinite, 446
- showRepetitiveDataLabels, 447
- suffix, 447
- textAttributes, 447

KDChart::DiagramObserver, 448

KDChart::DiagramObserver
- ~DiagramObserver, 449
- __pad0__, 451
- diagram, 449
- diagramAttributesChanged, 449
- diagramDataChanged, 449
- diagramDataHidden, 449
- DiagramObserver, 448
- Q_SLOTS, 451
- slotAttributesChanged, 450
- slotDataChanged, 450
- slotDataHidden, 450
- slotHeaderDataChanged, 451
- slotModelsChanged, 451

KDChart::FrameAttributes, 452

KDChart::FrameAttributes
- ~FrameAttributes, 452
- FrameAttributes, 452
- isVisible, 452
- operator!=, 452
- operator=, 452
- operator==, 452
- padding, 452
- pen, 452
- setPadding, 453
- setPen, 453
- setVisible, 453

KDChart::GlobalMeasureScaling, 454

KDChart::GlobalMeasureScaling
- ~GlobalMeasureScaling, 454
- currentFactors, 455
- GlobalMeasureScaling, 454
- instance, 455
- resetFactors, 455
- setFactors, 455

KDChart::GridAttributes, 457

KDChart::GridAttributes
- ~GridAttributes, 458
- adjustLowerBoundToGrid, 458
- adjustUpperBoundToGrid, 458
- GridAttributes, 458
- gridGranularitySequence, 458
- gridPen, 458
- gridStepWidth, 458
- gridSubStepWidth, 458
- isGridVisible, 458
- isSubGridVisible, 458
- operator!=, 458
- operator=, 458
- operator==, 459
- setAdjustBoundsToGrid, 459
- setGridGranularitySequence, 459
- setGridPen, 459
- setGridStepWidth, 459
- setGridSubStepWidth, 459
- setGridVisible, 459
- setSubGridPen, 459
- setSubGridVisible, 459
- setZeroLinePen, 459
- subGridPen, 459
- zeroLinePen, 459

KDChart::HeaderFooter, 460
- Footer, 462
- Header, 462

KDChart::HeaderFooter
- ~HeaderFooter, 462
- __pad0__, 477
- alignToReferencePoint, 463
- areaGeometry, 463
- autoReferenceArea, 463
- backgroundAttributes, 463
- clone, 463
- compare, 464
- expandingDirections, 464
- frameAttributes, 465
- geometry, 465
- getFrameLeadings, 465
- HeaderFooter, 462
- HeaderFooterType, 462
- innerRect, 465
- intersects, 466, 467
- isEmpty, 467
- maximumSize, 467
- minimumSize, 467
- mParent, 477
- mParentLayout, 477
- paint, 467
- paintAll, 468
- paintBackground, 469
- paintBackgroundAttributes, 469
- paintCtx, 470
- paintFrame, 470
- paintFrameAttributes, 471
- paintIntoRect, 471
- parentLayout, 471
- position, 472
- positionChanged, 472
- positionHasChanged, 472
- realFont, 472
- realFontSize, 472
- removeFromParentLayout, 472
- setAutoReferenceArea, 473
- setBackgroundAttributes, 473

  - setFrameAttributes, 473
  - setGeometry, 473
  - setParent, 474
  - setParentLayout, 474
  - setParentWidget, 474
  - setPosition, 474
  - setText, 475
  - setTextAttributes, 475
  - setType, 475
  - sizeHint, 476
  - sizeHintChanged, 476
  - text, 476
  - textAttributes, 477
  - type, 477
- KDChart::HorizontalLineLayoutItem, 479
- KDChart::HorizontalLineLayoutItem
  - expandingDirections, 480
  - geometry, 480
  - HorizontalLineLayoutItem, 479
  - isEmpty, 480
  - maximumSize, 480
  - minimumSize, 480
  - mParent, 483
  - mParentLayout, 483
  - paint, 480
  - paintAll, 481
  - paintCtx, 481
  - parentLayout, 481
  - removeFromParentLayout, 481
  - setGeometry, 482
  - setParentLayout, 482
  - setParentWidget, 482
  - sizeHint, 482
  - sizeHintChanged, 482
- KDChart::Legend, 496
  - ~Legend, 500
  - __pad0__, 532
  - activateTheLayout, 501
  - addDiagram, 501
  - alignment, 501
  - alignToReferencePoint, 502
  - areaGeometry, 502
  - backgroundAttributes, 502
  - brush, 502
  - brushes, 503
  - buildLegend, 503
  - clone, 507
  - compare, 507, 508
  - constDiagrams, 508
  - datasetCount, 509
  - diagram, 509
  - diagrams, 509
  - floatingPosition, 510
  - forceRebuild, 510
  - frameAttributes, 510
  - getFrameLeadings, 511
  - innerRect, 511
  - Legend, 500
  - LegendStyle, 500
  - legendStyle, 511
  - LinesOnly, 500
  - markerAttributes, 512
  - MarkersAndLines, 500
  - MarkersOnly, 500
  - minimumSizeHint, 512
  - needSizeHint, 512
  - orientation, 513
  - paint, 513
  - paintAll, 514
  - paintBackground, 514
  - paintBackgroundAttributes, 515
  - paintEvent, 516
  - paintFrame, 516
  - paintFrameAttributes, 516
  - paintIntoRect, 517
  - pen, 517
  - pens, 518
  - position, 518
  - positionHasChanged, 518
  - propertiesChanged, 518
  - Q_SLOTS, 532
  - referenceArea, 519
  - removeDiagram, 519
  - removeDiagrams, 519
  - replaceDiagram, 520
  - resetDiagram, 520
  - resetTexts, 520
  - resizeEvent, 521
  - resizeLayout, 521
  - setAlignment, 521
  - setBackgroundAttributes, 522
  - setBrush, 522
  - setBrushesFromDiagram, 522
  - setColor, 522
  - setDefaultColors, 523
  - setDiagram, 523
  - setFloatingPosition, 523
  - setFrameAttributes, 524
  - setLegendStyle, 525
  - setMarkerAttributes, 525
  - setNeedRebuild, 525
  - setOrientation, 525
  - setPen, 526
  - setPosition, 526
  - setRainbowColors, 526
  - setReferenceArea, 527
  - setShowLines, 527
  - setSpacing, 527

setSubduedColors, 528
setText, 528
setTextAttributes, 529
setTitleText, 529
setTitleTextAttributes, 529
setUseAutomaticMarkerSize, 529
setVisible, 530
showLines, 530
sizeHint, 530
spacing, 530
text, 531
textAttributes, 531
texts, 531
titleText, 531
titleTextAttributes, 532
useAutomaticMarkerSize, 532
KDChart::LineAttributes, 533
  MissingValuesAreBridged, 534
  MissingValuesHideSegments, 534
  MissingValuesPolicyIgnored, 534
  MissingValuesShownAsZero, 534
KDChart::LineAttributes
  ∼LineAttributes, 534
  displayArea, 534
  LineAttributes, 534
  MissingValuesPolicy, 533
  missingValuesPolicy, 534
  operator!=, 534
  operator=, 534
  operator==, 534
  setDisplayArea, 534
  setMissingValuesPolicy, 534
  setTransparency, 534
  transparency, 534
KDChart::LineDiagram, 536
  Normal, 541
  Percent, 541
  Stacked, 541
KDChart::LineDiagram
  ∼LineDiagram, 542
  __pad0__, 586
  addAxis, 542
  allowOverlappingDataValueTexts, 542
  antiAliasing, 543
  attributesModel, 543
  attributesModelRootIndex, 543
  axes, 544
  brush, 544
  calculateDataBoundaries, 545
  checkInvariants, 546
  clone, 547
  columnToIndex, 547
  compare, 547, 549
  coordinatePlane, 550
  dataBoundaries, 550
  dataChanged, 550
  dataHidden, 551
  datasetBrushes, 551
  datasetDimension, 551
  datasetLabels, 552
  datasetMarkers, 552
  datasetPens, 553
  dataValueAttributes, 553, 554
  doItemsLayout, 554
  getCellValues, 554
  horizontalOffset, 555
  indexAt, 555
  isHidden, 555, 556
  isIndexHidden, 556
  itemRowLabels, 556
  layoutPlanes, 557
  lineAttributes, 557, 558
  LineDiagram, 542
  LineType, 541
  modelsChanged, 558
  moveCursor, 558
  numberOfAbscissaSegments, 558
  numberOfOrdinateSegments, 558
  paint, 559
  paintDataValueText, 564
  paintDataValueTexts, 566
  paintEvent, 566
  paintMarker, 566, 567
  paintMarkers, 569
  pen, 569, 570
  percentMode, 570
  propertiesChanged, 570
  referenceDiagram, 570
  referenceDiagramOffset, 570
  resetLineAttributes, 571
  resize, 571
  resizeEvent, 571
  scrollTo, 571
  setAllowOverlappingDataValueTexts, 572
  setAntiAliasing, 572
  setAttributesModel, 572
  setAttributesModelRootIndex, 573
  setBrush, 573, 574
  setCoordinatePlane, 574
  setDataBoundariesDirty, 574
  setDatasetDimension, 575
  setDataValueAttributes, 575, 576
  setHidden, 576, 577
  setLineAttributes, 577, 578
  setModel, 578
  setPen, 578, 579
  setPercentMode, 579
  setReferenceDiagram, 579

- setRootIndex, 579
- setSelection, 580
- setThreeDLineAttributes, 580
- setType, 581
- takeAxis, 581
- threeDItemDepth, 582
- threeDLineAttributes, 582, 583
- type, 583
- update, 583
- useDefaultColors, 583
- useRainbowColors, 583
- usesExternalAttributesModel, 584
- useSubduedColors, 584
- valueForCell, 584
- valueForCellTesting, 585
- verticalOffset, 585
- visualRect, 585
- visualRegionForSelection, 585

KDChart::LineLayoutItem, 587

KDChart::LineLayoutItem
- expandingDirections, 588
- geometry, 588
- isEmpty, 588
- LineLayoutItem, 588
- maximumSize, 588
- minimumSize, 588
- mParent, 592
- mParentLayout, 592
- paint, 589
- paintAll, 589
- paintCtx, 589
- paintIntoRect, 589
- parentLayout, 590
- removeFromParentLayout, 590
- setGeometry, 590
- setParentLayout, 590
- setParentWidget, 591
- sizeHint, 591
- sizeHintChanged, 591

KDChart::LineWithMarkerLayoutItem, 593

KDChart::LineWithMarkerLayoutItem
- expandingDirections, 594
- geometry, 594
- isEmpty, 594
- LineWithMarkerLayoutItem, 593
- maximumSize, 594
- minimumSize, 594
- mParent, 597
- mParentLayout, 597
- paint, 595
- paintAll, 595
- paintCtx, 595
- parentLayout, 596
- removeFromParentLayout, 596
- setGeometry, 596
- setParentLayout, 596
- setParentWidget, 596
- sizeHint, 597
- sizeHintChanged, 597

KDChart::MarkerAttributes, 599
- Marker1Pixel, 600
- Marker4Pixels, 600
- MarkerCircle, 600
- MarkerCross, 600
- MarkerDiamond, 600
- MarkerFastCross, 600
- MarkerRing, 600
- MarkerSquare, 600

KDChart::MarkerAttributes
- ~MarkerAttributes, 600
- isVisible, 600
- MarkerAttributes, 600
- markerColor, 600
- markerSize, 600
- MarkerStyle, 600
- markerStyle, 600
- MarkerStylesMap, 599
- markerStylesMap, 601
- operator!=, 601
- operator=, 601
- operator==, 601
- pen, 601
- setMarkerColor, 601
- setMarkerSize, 601
- setMarkerStyle, 601
- setMarkerStylesMap, 601
- setPen, 601
- setVisible, 601

KDChart::MarkerLayoutItem, 602

KDChart::MarkerLayoutItem
- expandingDirections, 603
- geometry, 603
- isEmpty, 603
- MarkerLayoutItem, 603
- maximumSize, 603
- minimumSize, 603
- mParent, 607
- mParentLayout, 607
- paint, 604
- paintAll, 604
- paintCtx, 604
- paintIntoRect, 604
- parentLayout, 605
- removeFromParentLayout, 605
- setGeometry, 606
- setParentLayout, 606
- setParentWidget, 606
- sizeHint, 606

- sizeHintChanged, 606

KDChart::Measure, 608
- calculatedValue, 609, 610
- calculationMode, 610
- Measure, 609
- operator!=, 611
- operator=, 611
- operator==, 611
- referenceArea, 611
- referenceOrientation, 611
- setAbsoluteValue, 612
- setCalculationMode, 612
- setReferenceArea, 612
- setReferenceOrientation, 612
- setRelativeMode, 612
- setValue, 613
- sizeOfArea, 613
- value, 613

KDChart::PaintContext, 615

KDChart::PaintContext
- ~PaintContext, 615
- coordinatePlane, 615
- PaintContext, 615
- painter, 615
- rectangle, 615
- setCoordinatePlane, 615
- setPainter, 615
- setRectangle, 616

KDChart::Palette, 617
- ~Palette, 618
- __pad0__, 619
- addBrush, 618
- defaultPalette, 618
- getBrush, 618
- isValid, 618
- operator=, 618
- Palette, 618
- rainbowPalette, 618
- removeBrush, 618
- size, 619
- subduedPalette, 619

KDChart::PieAttributes, 620

KDChart::PieAttributes
- ~PieAttributes, 620
- explode, 621
- explodeFactor, 621
- operator!=, 621
- operator=, 621
- operator==, 621
- PieAttributes, 620
- setExplode, 622
- setExplodeFactor, 622

KDChart::PieDiagram, 623

KDChart::PieDiagram
- ~PieDiagram, 628
- __pad0__, 664
- allowOverlappingDataValueTexts, 628
- antiAliasing, 628
- attributesModel, 629
- attributesModelRootIndex, 629
- brush, 629, 630
- calculateDataBoundaries, 630
- checkInvariants, 631
- clone, 631
- columnCount, 632
- columnToIndex, 632
- compare, 632
- coordinatePlane, 634
- dataBoundaries, 634
- dataChanged, 634
- dataHidden, 635
- datasetBrushes, 635
- datasetDimension, 635
- datasetLabels, 636
- datasetMarkers, 636
- datasetPens, 637
- dataValueAttributes, 637, 638
- doItemsLayout, 638
- granularity, 638
- horizontalOffset, 639
- indexAt, 639
- isHidden, 639, 640
- isIndexHidden, 640
- itemRowLabels, 640
- modelsChanged, 641
- moveCursor, 641
- numberOfGridRings, 641
- numberOfValuesPerDataset, 641
- paint, 641
- paintDataValueText, 644
- paintDataValueTexts, 646
- paintEvent, 646
- paintMarker, 646, 647
- paintMarkers, 648
- pen, 649
- percentMode, 650
- pieAttributes, 650
- PieDiagram, 628
- polarCoordinatePlane, 651
- propertiesChanged, 651
- resize, 651
- resizeEvent, 651
- scrollTo, 651
- setAllowOverlappingDataValueTexts, 651
- setAntiAliasing, 652
- setAttributesModel, 652
- setAttributesModelRootIndex, 653
- setBrush, 653

- setCoordinatePlane, 654
- setDataBoundariesDirty, 654
- setDatasetDimension, 654
- setDataValueAttributes, 655
- setGranularity, 656
- setHidden, 656, 657
- setModel, 657
- setPen, 658
- setPercentMode, 658
- setPieAttributes, 659
- setRootIndex, 659
- setSelection, 659
- setStartPosition, 659
- setThreeDPieAttributes, 660
- startPosition, 660
- threeDPieAttributes, 661
- update, 661
- useDefaultColors, 662
- useRainbowColors, 662
- usesExternalAttributesModel, 662
- useSubduedColors, 662
- valueForCell, 663
- valueTotals, 663
- verticalOffset, 663
- visualRect, 664
- visualRegionForSelection, 664

KDChart::PolarCoordinatePlane, 665
- Linear, 669
- Logarithmic, 669

KDChart::PolarCoordinatePlane
- ~PolarCoordinatePlane, 669
- __pad0__, 696
- addDiagram, 670
- alignToReferencePoint, 670
- angleUnit, 670
- areaGeometry, 671
- AxesCalcMode, 669
- backgroundAttributes, 671
- bottomOverlap, 671
- compare, 672
- CoordinateTransformationList, 669
- diagram, 672
- diagrams, 672, 673
- expandingDirections, 673
- frameAttributes, 673
- geometry, 673
- getDataDimensionsList, 674
- getFrameLeadings, 674
- globalGridAttributes, 674
- gridAttributes, 675
- gridDimensionsList, 675
- hasOwnGridAttributes, 676
- innerRect, 677
- isEmpty, 677
- isVisiblePoint, 677
- layoutDiagrams, 677
- layoutPlanes, 678
- leftOverlap, 679
- maximumSize, 679
- minimumSize, 679
- minimumSizeHint, 680
- mousePressEvent, 680
- mParent, 696
- mParentLayout, 697
- needLayoutPlanes, 680
- needRelayout, 680
- needUpdate, 680
- paint, 680
- paintAll, 681
- paintBackground, 682
- paintBackgroundAttributes, 682
- paintCtx, 683
- paintEvent, 683
- paintFrame, 683
- paintFrameAttributes, 683
- paintIntoRect, 684
- parent, 684
- parentLayout, 685
- PolarCoordinatePlane, 669
- positionHasChanged, 685
- propertiesChanged, 685
- Q_SLOTS, 697
- radiusUnit, 685
- referenceCoordinatePlane, 686
- relayout, 686
- removeFromParentLayout, 686
- replaceDiagram, 687
- resetGridAttributes, 687
- resizeEvent, 688
- rightOverlap, 688
- setBackgroundAttributes, 688
- setFrameAttributes, 688
- setGeometry, 689
- setGlobalGridAttributes, 689
- setGridAttributes, 689
- setGridNeedsRecalculate, 690
- setParent, 690
- setParentLayout, 691
- setParentWidget, 691
- setReferenceCoordinatePlane, 691
- setStartPosition, 691
- setZoomCenter, 692
- setZoomFactorX, 692
- setZoomFactorY, 692
- sizeHint, 693
- sizeHintChanged, 693
- sizePolicy, 693
- startPosition, 693

takeDiagram, 694
topOverlap, 694
translate, 695
translatePolar, 695
zoomCenter, 695
zoomFactorX, 696
zoomFactorY, 696
KDChart::PolarDiagram, 698
KDChart::PolarDiagram
∼PolarDiagram, 703
__pad0__, 735
allowOverlappingDataValueTexts, 703
antiAliasing, 703
attributesModel, 703
attributesModelRootIndex, 704
brush, 704, 705
calculateDataBoundaries, 705
checkInvariants, 705
clone, 706
closeDatasets, 706
columnCount, 706
columnToIndex, 707
compare, 707
coordinatePlane, 708
dataBoundaries, 709
dataChanged, 709
dataHidden, 710
datasetBrushes, 710
datasetDimension, 710
datasetLabels, 711
datasetMarkers, 711
datasetPens, 711
dataValueAttributes, 712, 713
doItemsLayout, 713
horizontalOffset, 713
indexAt, 713
isHidden, 713, 714
isIndexHidden, 714
itemRowLabels, 714
modelsChanged, 715
moveCursor, 715
numberOfGridRings, 715
numberOfValuesPerDataset, 715
paint, 716
paintDataValueText, 716
paintDataValueTexts, 718
paintEvent, 718
paintMarker, 718, 719
paintMarkers, 721
paintPolarMarkers, 721
pen, 721, 722
percentMode, 722
polarCoordinatePlane, 722
PolarDiagram, 703
propertiesChanged, 722
resize, 723
resizeEvent, 723
rotateCircularLabels, 723
scrollTo, 723
setAllowOverlappingDataValueTexts, 723
setAntiAliasing, 724
setAttributesModel, 724
setAttributesModelRootIndex, 725
setBrush, 725
setCloseDatasets, 726
setCoordinatePlane, 726
setDataBoundariesDirty, 726
setDatasetDimension, 727
setDataValueAttributes, 727
setHidden, 728, 729
setModel, 729
setPen, 729, 730
setPercentMode, 730
setRootIndex, 730
setRotateCircularLabels, 731
setSelection, 731
setShowDelimitersAtPosition, 731
setShowLabelsAtPosition, 731
setZeroDegreePosition, 731
showDelimitersAtPosition, 732
showLabelsAtPosition, 732
update, 732
useDefaultColors, 732
useRainbowColors, 732
usesExternalAttributesModel, 733
useSubduedColors, 733
valueForCell, 733
valueTotals, 734
verticalOffset, 734
visualRect, 734
visualRegionForSelection, 734
zeroDegreePosition, 734
KDChart::Position, 736
Center, 743
East, 743
ExcludeCenter, 737
Floating, 743
fromName, 738
IncludeCenter, 737
isCorner, 739
isEastSide, 739
isFloating, 739
isNorthSide, 739
isPole, 740
isSouthSide, 740
isUnknown, 740
isWestSide, 740
name, 740

- names, 741
- North, 743
- NorthEast, 743
- NorthWest, 743
- operator!=, 741
- operator==, 741, 742
- Option, 737
- Position, 738
- printableName, 742
- printableNames, 742
- South, 743
- SouthEast, 743
- SouthWest, 743
- Unknown, 744
- value, 742
- West, 744

KDChart::PositionPoints, 745

KDChart::PositionPoints
- isNull, 747
- mPositionCenter, 747
- mPositionEast, 747
- mPositionNorth, 747
- mPositionNorthEast, 748
- mPositionNorthWest, 748
- mPositionSouth, 748
- mPositionSouthEast, 748
- mPositionSouthWest, 748
- mPositionUnknown, 748
- mPositionWest, 748
- point, 747
- PositionPoints, 745, 746

KDChart::RelativePosition, 766

KDChart::RelativePosition
- ∼RelativePosition, 767
- alignment, 767
- calculatedPoint, 767
- horizontalPadding, 767
- operator!=, 767
- operator=, 768
- operator==, 768
- referenceArea, 768
- referencePoint, 768
- referencePoints, 768
- referencePosition, 768
- RelativePosition, 767
- resetReferencePosition, 768
- rotation, 768
- setAlignment, 769
- setHorizontalPadding, 769
- setReferenceArea, 769
- setReferencePoints, 769
- setReferencePosition, 770
- setRotation, 770
- setVerticalPadding, 770
- verticalPadding, 770

KDChart::RingDiagram, 771

KDChart::RingDiagram
- ∼RingDiagram, 776
- __pad0__, 810
- allowOverlappingDataValueTexts, 776
- antiAliasing, 776
- attributesModel, 777
- attributesModelRootIndex, 777
- brush, 777, 778
- calculateDataBoundaries, 778
- checkInvariants, 779
- clone, 779
- columnCount, 779
- columnToIndex, 780
- compare, 780
- coordinatePlane, 781
- dataBoundaries, 782
- dataChanged, 782
- dataHidden, 783
- datasetBrushes, 783
- datasetDimension, 783
- datasetLabels, 784
- datasetMarkers, 784
- datasetPens, 784
- dataValueAttributes, 785, 786
- doItemsLayout, 786
- granularity, 786
- horizontalOffset, 786
- indexAt, 787
- isHidden, 787
- isIndexHidden, 788
- itemRowLabels, 788
- modelsChanged, 788
- moveCursor, 788
- numberOfGridRings, 789
- numberOfValuesPerDataset, 789
- paint, 789
- paintDataValueText, 790
- paintDataValueTexts, 791
- paintEvent, 791
- paintMarker, 792
- paintMarkers, 794
- pen, 794, 795
- percentMode, 795
- pieAttributes, 795, 796
- polarCoordinatePlane, 796
- propertiesChanged, 796
- relativeThickness, 796
- resize, 796
- resizeEvent, 797
- RingDiagram, 776
- scrollTo, 797
- setAllowOverlappingDataValueTexts, 797

- setAntiAliasing, 797
- setAttributesModel, 797
- setAttributesModelRootIndex, 798
- setBrush, 798, 799
- setCoordinatePlane, 799
- setDataBoundariesDirty, 799
- setDatasetDimension, 800
- setDataValueAttributes, 800, 801
- setGranularity, 801
- setHidden, 801, 802
- setModel, 803
- setPen, 803
- setPercentMode, 804
- setPieAttributes, 804
- setRelativeThickness, 804
- setRootIndex, 805
- setSelection, 805
- setStartPosition, 805
- setThreeDPieAttributes, 805, 806
- startPosition, 806
- threeDPieAttributes, 806, 807
- update, 807
- useDefaultColors, 807
- useRainbowColors, 807
- usesExternalAttributesModel, 808
- useSubduedColors, 808
- valueForCell, 808
- valueTotals, 809
- verticalOffset, 809
- visualRect, 809
- visualRegionForSelection, 809

KDChart::SignalCompressor, 811

KDChart::SignalCompressor
- __pad0__, 811
- Q_SLOTS, 811, 812
- SignalCompressor, 811

KDChart::TextArea, 813

KDChart::TextArea
- ~TextArea, 815
- __pad0__, 828
- alignToReferencePoint, 815
- areaGeometry, 815
- autoReferenceArea, 816
- backgroundAttributes, 816
- compare, 816
- expandingDirections, 816
- frameAttributes, 817
- geometry, 817
- getFrameLeadings, 817
- innerRect, 817
- intersects, 818, 819
- isEmpty, 819
- maximumSize, 819
- minimumSize, 819
- mParent, 828
- mParentLayout, 828
- paint, 819
- paintAll, 820
- paintBackground, 821
- paintBackgroundAttributes, 821
- paintCtx, 822
- paintFrame, 822
- paintFrameAttributes, 823
- paintIntoRect, 823
- parentLayout, 823
- positionHasChanged, 824
- realFont, 824
- realFontSize, 824
- removeFromParentLayout, 824
- setAutoReferenceArea, 824
- setBackgroundAttributes, 825
- setFrameAttributes, 825
- setGeometry, 825
- setParentLayout, 825
- setParentWidget, 826
- setText, 826
- setTextAttributes, 826
- sizeHint, 827
- sizeHintChanged, 827
- text, 827
- TextArea, 815
- textAttributes, 828

KDChart::TextAttributes, 829

KDChart::TextAttributes
- ~TextAttributes, 830
- autoRotate, 830
- autoShrink, 830
- calculatedFont, 830
- calculatedFontSize, 830
- font, 831
- fontSize, 831
- hasAbsoluteFontSize, 831
- isVisible, 831
- minimalFontSize, 831
- operator!=, 831
- operator=, 832
- operator==, 832
- pen, 832
- rotation, 832
- setAutoRotate, 832
- setAutoShrink, 832
- setFont, 832
- setFontSize, 832
- setMinimalFontSize, 833
- setPen, 833
- setRotation, 833
- setVisible, 833
- TextAttributes, 830

KDChart::TextLayoutItem, 834
KDChart::TextLayoutItem
  autoReferenceArea, 836
  expandingDirections, 836
  geometry, 836
  intersects, 836, 837
  isEmpty, 837
  maximumSize, 837
  minimumSize, 838
  mParent, 843
  mParentLayout, 843
  paint, 838
  paintAll, 839
  paintCtx, 839
  parentLayout, 839
  realFont, 839
  realFontSize, 839
  removeFromParentLayout, 840
  setAutoReferenceArea, 840
  setGeometry, 840
  setParentLayout, 840
  setParentWidget, 841
  setText, 841
  setTextAttributes, 841
  sizeHint, 842
  sizeHintChanged, 842
  text, 842
  textAttributes, 842
  TextLayoutItem, 835
KDChart::ThreeDBarAttributes, 844
KDChart::ThreeDBarAttributes
  ∼ThreeDBarAttributes, 844
  angle, 845
  depth, 845
  isEnabled, 845
  operator!=, 845
  operator=, 846
  operator==, 846
  setAngle, 846
  setDepth, 847
  setEnabled, 847
  setUseShadowColors, 847
  ThreeDBarAttributes, 844
  useShadowColors, 847
  validDepth, 847
KDChart::ThreeDLineAttributes, 849
KDChart::ThreeDLineAttributes
  ∼ThreeDLineAttributes, 849
  depth, 850
  isEnabled, 850
  lineXRotation, 850
  lineYRotation, 850
  operator!=, 851
  operator=, 851
  operator==, 851
  setDepth, 852
  setEnabled, 852
  setLineXRotation, 852
  setLineYRotation, 852
  ThreeDLineAttributes, 849
  validDepth, 852
KDChart::ThreeDPieAttributes, 854
KDChart::ThreeDPieAttributes
  ∼ThreeDPieAttributes, 854
  depth, 855
  isEnabled, 855
  operator!=, 855
  operator=, 855
  operator==, 856
  setDepth, 856
  setEnabled, 856
  setUseShadowColors, 857
  ThreeDPieAttributes, 854
  useShadowColors, 857
  validDepth, 857
KDChart::VerticalLineLayoutItem, 858
KDChart::VerticalLineLayoutItem
  expandingDirections, 859
  geometry, 859
  isEmpty, 859
  maximumSize, 859
  minimumSize, 859
  mParent, 862
  mParentLayout, 862
  paint, 859
  paintAll, 860
  paintCtx, 860
  parentLayout, 860
  removeFromParentLayout, 860
  setGeometry, 861
  setParentLayout, 861
  setParentWidget, 861
  sizeHint, 861
  sizeHintChanged, 861
  VerticalLineLayoutItem, 858
KDChart::Widget, 863
  ∼Widget, 866
  addHeaderFooter, 867
  addLegend, 867, 868
  allHeadersFooters, 868
  allLegends, 868
  Bar, 866
  barDiagram, 868
  bottom, 877
  ChartType, 866
  coordinatePlane, 868
  diagram, 869
  firstHeaderFooter, 869

  globalLeadingBottom, 869
  globalLeadingLeft, 869
  globalLeadingRight, 870
  globalLeadingTop, 870
  legend, 870
  Line, 866
  lineDiagram, 870
  Normal, 866
  NoType, 866
  Percent, 866
  Pie, 866
  pieDiagram, 871
  Polar, 866
  polarDiagram, 871
  Q_SLOTS, 877
  replaceHeaderFooter, 871
  replaceLegend, 872
  resetData, 872
  right, 877
  Ring, 866
  ringDiagram, 872
  Rows, 866
  setDataCell, 872, 873
  setDataset, 873
  setGlobalLeadingBottom, 874
  setGlobalLeadingLeft, 874
  setGlobalLeadingRight, 874
  setGlobalLeadingTop, 875
  setSubType, 875
  Stacked, 866
  SubType, 866
  subType, 876, 877
  takeHeaderFooter, 876
  takeLegend, 876
  top, 878
  type, 876
  Widget, 866
KDChart::ZoomParameters, 879
KDChart::ZoomParameters
  center, 879
  setCenter, 879
  xCenter, 880
  xFactor, 880
  yCenter, 880
  yFactor, 880
  ZoomParameters, 879
KDCHART_DATA_VALUE_AUTO_DIGITS
  KDChartDataValueAttributes.cpp, 927
KDCHART_DECLARE_DERIVED_DIAGRAM
  KDChartGlobal.h, 936
KDCHART_DECLARE_PRIVATE_BASE_POLYMORPHIC
  KDChartGlobal.h, 937
KDCHART_DECLARE_PRIVATE_BASE_POLYMORPHIC_QWIDGET
  KDChartGlobal.h, 937
KDCHART_DECLARE_PRIVATE_BASE_VALUE
  KDChartGlobal.h, 937
KDCHART_DECLARE_PRIVATE_DERIVED
  KDChartGlobal.h, 938
KDCHART_DECLARE_PRIVATE_DERIVED_PARENT
  KDChartGlobal.h, 938
KDCHART_DECLARE_PRIVATE_DERIVED_QWIDGET
  KDChartGlobal.h, 938
KDCHART_DECLARE_SWAP_BASE
  KDChartGlobal.h, 938
KDCHART_DECLARE_SWAP_DERIVED
  KDChartGlobal.h, 938
KDCHART_DECLARE_SWAP_SPECIALISATION
  KDChartGlobal.h, 938
KDCHART_DECLARE_SWAP_SPECIALISATION_DERIVED
  KDChartGlobal.h, 939
KDCHART_DERIVED_PRIVATE_FOOTER
  KDChartGlobal.h, 939
KDCHART_IMPL_DERIVED_DIAGRAM
  KDChartGlobal.h, 939
KDCHART_IMPL_DERIVED_PLANE
  KDChartGlobal.h, 939
KDChartAbstractArea.cpp, 881
KDChartAbstractArea.cpp
  d, 881
KDChartAbstractArea.h, 885
KDChartAbstractAreaBase.cpp, 886
KDChartAbstractAreaBase.cpp
  d, 886
KDChartAbstractAreaBase.h, 887
KDChartAbstractAreaWidget.cpp, 888
KDChartAbstractAreaWidget.cpp
  d, 888
KDChartAbstractAreaWidget.h, 889
KDChartAbstractAxis.cpp, 890
KDChartAbstractAxis.cpp
  d, 890
KDChartAbstractAxis.h, 891
KDChartAbstractCartesianDiagram.cpp, 892
KDChartAbstractCartesianDiagram.cpp
  d, 892
KDChartAbstractCartesianDiagram.h, 893
KDChartAbstractCoordinatePlane.cpp, 894
KDChartAbstractCoordinatePlane.cpp
  d, 894
KDChartAbstractCoordinatePlane.h, 895

KDChartAbstractDiagram.cpp, 896
KDChartAbstractDiagram.cpp
  d, 896
KDChartAbstractDiagram.h, 897
KDChartAbstractPieDiagram.cpp, 898
KDChartAbstractPieDiagram.cpp
  d, 898
KDChartAbstractPieDiagram.h, 899
KDChartAbstractPolarDiagram.cpp, 900
KDChartAbstractPolarDiagram.cpp
  d, 900
KDChartAbstractPolarDiagram.h, 901
KDChartAbstractProxyModel.cpp, 902
KDChartAbstractProxyModel.h, 903
KDChartAbstractThreeDAttributes.cpp, 904
KDChartAbstractThreeDAttributes.cpp
  d, 904
  operator<<, 904
KDChartAbstractThreeDAttributes.h, 905
KDChartAbstractThreeDAttributes.h
  operator<<, 905
KDChartAttributesModel.cpp, 906
KDChartAttributesModel.h, 907
KDChartBackgroundAttributes.cpp, 908
KDChartBackgroundAttributes.cpp
  d, 908
  operator<<, 908
KDChartBackgroundAttributes.h, 909
KDChartBackgroundAttributes.h
  operator<<, 909
  Q_DECLARE_TYPEINFO, 909
KDChartBarAttributes.cpp, 910
KDChartBarAttributes.cpp
  d, 910
KDChartBarAttributes.h, 911
KDChartBarDiagram.cpp, 912
KDChartBarDiagram.cpp
  d, 912
KDChartBarDiagram.h, 913
KDChartCartesianAxis.cpp, 914
KDChartCartesianAxis.cpp
  calculateNextLabel, 914
  calculateOverlap, 915
  d, 914
  referenceDiagramIsBarDiagram, 915
KDChartCartesianAxis.h, 916
KDChartCartesianCoordinatePlane.cpp, 917
KDChartCartesianCoordinatePlane.cpp
  d, 917
KDChartCartesianCoordinatePlane.h, 918
KDChartChart.cpp, 919
KDChartChart.cpp
  ADD_AUTO_SPACER_IF_NEEDED, 920
  ADD_VBOX_WITH_LEGENDS, 920
  d, 920
  findOrCreateHBoxLayoutByObjectName, 920
  findOrCreateLayoutByObjectName, 920
  findOrCreateVBoxLayoutByObjectName, 921
  SET_ALL_MARGINS_TO_ZERO, 920
KDChartChart.h, 922
KDChartDatasetProxyModel.cpp, 923
KDChartDatasetProxyModel.h, 924
KDChartDatasetSelector.cpp, 925
KDChartDatasetSelector.h, 926
KDChartDataValueAttributes.cpp, 927
KDChartDataValueAttributes.cpp
  d, 927
  KDCHART_DATA_VALUE_AUTO_DIGITS, 927
  operator<<, 927
KDChartDataValueAttributes.h, 929
KDChartDataValueAttributes.h
  operator<<, 929
  Q_DECLARE_TYPEINFO, 930
KDChartDiagramObserver.cpp, 931
KDChartDiagramObserver.h, 932
KDChartEnums, 484
  GranularitySequence_10_20, 486
  GranularitySequence_10_50, 486
  GranularitySequence_125_25, 487
  GranularitySequence_25_50, 487
  GranularitySequenceIrregular, 487
  LayoutJustOverwrite, 489
  LayoutPolicyRotate, 489
  LayoutPolicyShiftHorizontally, 489
  LayoutPolicyShiftVertically, 489
  LayoutPolicyShrinkFontSize, 489
  MeasureCalculationModeAbsolute, 487
  MeasureCalculationModeAuto, 487
  MeasureCalculationModeAutoArea, 487
  MeasureCalculationModeAutoOrientation, 487
  MeasureCalculationModeRelative, 487
  MeasureOrientationAuto, 488
  MeasureOrientationHorizontal, 488
  MeasureOrientationMaximum, 488
  MeasureOrientationMinimum, 488
  MeasureOrientationVertical, 488
  PositionCenter, 488
  PositionEast, 489
  PositionFloating, 489
  PositionNorth, 489
  PositionNorthEast, 489
  PositionNorthWest, 489
  PositionSouth, 489
  PositionSouthEast, 489

  PositionSouthWest, 489
  PositionUnknown, 488
  PositionWest, 489
KDChartEnums
  GranularitySequence, 486
  granularitySequenceToString, 490
  layoutPolicyToString, 490
  MeasureCalculationMode, 487
  measureCalculationModeToString, 490
  MeasureOrientation, 487
  measureOrientationToString, 491
  PositionValue, 488
  stringToGranularitySequence, 491
  stringToLayoutPolicy, 492
  stringToMeasureCalculationMode, 492
  stringToMeasureOrientation, 493
  TextLayoutPolicy, 489
KDChartEnums.h, 933
KDChartFrameAttributes.cpp, 934
KDChartFrameAttributes.cpp
  d, 934
  operator<<, 934
KDChartFrameAttributes.h, 935
KDChartFrameAttributes.h
  operator<<, 935
  Q_DECLARE_TYPEINFO, 935
KDChartGlobal.h, 936
KDChartGlobal.h
  __kdab__dereference_for_methodcall, 940
  KDAB_SET_OBJECT_NAME, 936
  KDCHART_DECLARE_DERIVED_DIAGRAM, 936
  KDCHART_DECLARE_PRIVATE_BASE_POLYMORPHIC, 937
  KDCHART_DECLARE_PRIVATE_BASE_POLYMORPHIC_QWIDGET, 937
  KDCHART_DECLARE_PRIVATE_BASE_VALUE, 937
  KDCHART_DECLARE_PRIVATE_DERIVED, 938
  KDCHART_DECLARE_PRIVATE_DERIVED_PARENT, 938
  KDCHART_DECLARE_PRIVATE_DERIVED_QWIDGET, 938
  KDCHART_DECLARE_SWAP_BASE, 938
  KDCHART_DECLARE_SWAP_DERIVED, 938
  KDCHART_DECLARE_SWAP_SPECIALISATION, 938
  KDCHART_DECLARE_SWAP_SPECIALISATION_DERIVED, 939
  KDCHART_DERIVED_PRIVATE_FOOTER, 939
  KDCHART_IMPL_DERIVED_DIAGRAM, 939
  KDCHART_IMPL_DERIVED_PLANE, 939
KDChartGridAttributes.cpp, 941
KDChartGridAttributes.cpp
  d, 941
  operator<<, 941
KDChartGridAttributes.h, 942
KDChartGridAttributes.h
  operator<<, 942
  Q_DECLARE_TYPEINFO, 942
KDChartHeaderFooter.cpp, 943
KDChartHeaderFooter.cpp
  d, 943
KDChartHeaderFooter.h, 944
KDChartLayoutItems.cpp, 945
KDChartLayoutItems.cpp
  PI, 945
  rotatedPoint, 946
  rotatedRect, 946
  updateCommonBrush, 946
KDChartLayoutItems.h, 948
KDChartLegend.cpp, 949
KDChartLegend.cpp
  d, 949
KDChartLegend.h, 950
KDChartLineAttributes.cpp, 951
KDChartLineAttributes.cpp
  d, 951
  operator<<, 951
KDChartLineAttributes.h, 952
KDChartLineAttributes.h
  operator<<, 952
  Q_DECLARE_TYPEINFO, 952
KDChartLineDiagram.cpp, 953
KDChartLineDiagram.cpp
  d, 953
KDChartLineDiagram.h, 954
KDChartMarkerAttributes.cpp, 955
KDChartMarkerAttributes.cpp
  d, 955
  operator<<, 955
KDChartMarkerAttributes.h, 956
KDChartMarkerAttributes.h
  operator<<, 956
  Q_DECLARE_TYPEINFO, 956
KDChartMeasure.cpp, 957
KDChartMeasure.cpp
  operator<<, 957
KDChartMeasure.h, 958
KDChartMeasure.h

  - operator<<, 958
- KDChartPaintContext.cpp, 959
- KDChartPaintContext.cpp
  - d, 959
- KDChartPaintContext.h, 960
- KDChartPalette.cpp, 961
- KDChartPalette.cpp
  - d, 961
  - makeDefaultPalette, 961
  - makeRainbowPalette, 961
  - makeSubduedPalette, 962
- KDChartPalette.h, 963
- KDChartPieAttributes.cpp, 964
- KDChartPieAttributes.cpp
  - d, 964
  - operator<<, 964
- KDChartPieAttributes.h, 965
- KDChartPieAttributes.h
  - operator<<, 965
  - Q_DECLARE_TYPEINFO, 965
- KDChartPieDiagram.cpp, 966
- KDChartPieDiagram.cpp
  - buildReferenceRect, 966
  - d, 966
- KDChartPieDiagram.h, 968
- KDChartPolarCoordinatePlane.cpp, 969
- KDChartPolarCoordinatePlane.cpp
  - d, 969
- KDChartPolarCoordinatePlane.h, 970
- KDChartPolarDiagram.cpp, 971
- KDChartPolarDiagram.cpp
  - d, 971
- KDChartPolarDiagram.h, 972
- KDChartPosition.cpp, 973
- KDChartPosition.cpp
  - maxPositionValue, 974
  - operator<<, 973
  - staticPositionCenter, 974
  - staticPositionEast, 974
  - staticPositionFloating, 974
  - staticPositionNames, 974
  - staticPositionNorth, 974
  - staticPositionNorthEast, 974
  - staticPositionNorthWest, 974
  - staticPositionSouth, 975
  - staticPositionSouthEast, 975
  - staticPositionSouthWest, 975
  - staticPositionUnknown, 975
  - staticPositionWest, 975
- KDChartPosition.h, 976
- KDChartPosition.h
  - operator<<, 976
  - Q_DECLARE_TYPEINFO, 976
- KDChartRelativePosition.cpp, 977
- KDChartRelativePosition.cpp
  - d, 977
  - operator<<, 977
- KDChartRelativePosition.h, 978
- KDChartRelativePosition.h
  - operator<<, 978
  - Q_DECLARE_TYPEINFO, 978
- KDChartRingDiagram.cpp, 979
- KDChartRingDiagram.cpp
  - d, 979
- KDChartRingDiagram.h, 980
- KDChartSignalCompressor.cpp, 981
- KDChartSignalCompressor.h, 982
- KDChartTextArea.cpp, 983
- KDChartTextArea.h, 984
- KDChartTextAttributes.cpp, 985
- KDChartTextAttributes.cpp
  - d, 985
  - operator<<, 985
- KDChartTextAttributes.h, 986
- KDChartTextAttributes.h
  - operator<<, 986
  - Q_DECLARE_TYPEINFO, 986
- KDChartTextLabelCache.cpp, 987
- KDChartTextLabelCache.cpp
  - DUMP_CACHE_STATS, 987
  - HitCount, 988
  - INC_HIT_COUNT, 987
  - INC_MISS_COUNT, 987
  - MissCount, 988
- KDChartTextLabelCache.h, 989
- KDChartThreeDBarAttributes.cpp, 990
- KDChartThreeDBarAttributes.cpp
  - d, 990
  - operator<<, 990
- KDChartThreeDBarAttributes.h, 991
- KDChartThreeDBarAttributes.h
  - operator<<, 991
  - Q_DECLARE_TYPEINFO, 991
- KDChartThreeDLineAttributes.cpp, 992
- KDChartThreeDLineAttributes.cpp
  - d, 992
  - operator<<, 992
- KDChartThreeDLineAttributes.h, 993
- KDChartThreeDLineAttributes.h
  - operator<<, 993
  - Q_DECLARE_TYPEINFO, 993
- KDChartThreeDPieAttributes.cpp, 994
- KDChartThreeDPieAttributes.cpp
  - d, 994
  - operator<<, 994
- KDChartThreeDPieAttributes.h, 995
- KDChartThreeDPieAttributes.h
  - operator<<, 995

  Q_DECLARE_TYPEINFO, 995
KDChartWidget.cpp, 996
KDChartWidget.cpp
  d, 996
  isCartesian, 997
  isPolar, 997
  SET_SUB_TYPE, 996
  TEST_SUB_TYPE, 997
KDChartWidget.h, 998
KDChartZoomParameters.h, 999
KDTextDocument, 494
  KDTextDocument, 494
KDTextDocument
  ∼KDTextDocument, 494
  KDTextDocument, 494
  minimumSizeHint, 495
  sizeHint, 495
KDTextDocument.cpp, 1000
KDTextDocument.h, 1001

labels
  KDChart::AbstractAxis, 68
  KDChart::CartesianAxis, 337
layoutDiagrams
  KDChart::AbstractCoordinatePlane, 125
  KDChart::CartesianCoordinatePlane, 383
  KDChart::PolarCoordinatePlane, 677
LayoutJustOverwrite
  KDChartEnums, 489
layoutPlanes
  KDChart::AbstractCartesianDiagram, 96
  KDChart::AbstractCoordinatePlane, 125
  KDChart::BarDiagram, 300
  KDChart::CartesianAxis, 337
  KDChart::CartesianCoordinatePlane, 384
  KDChart::LineDiagram, 557
  KDChart::PolarCoordinatePlane, 678
LayoutPolicyRotate
  KDChartEnums, 489
LayoutPolicyShiftHorizontally
  KDChartEnums, 489
LayoutPolicyShiftVertically
  KDChartEnums, 489
LayoutPolicyShrinkFontSize
  KDChartEnums, 489
layoutPolicyToString
  KDChartEnums, 490
Left
  KDChart::CartesianAxis, 329
leftOverlap
  KDChart::AbstractArea, 37
  KDChart::AbstractAxis, 69
  KDChart::AbstractCoordinatePlane, 126
  KDChart::CartesianAxis, 338
  KDChart::CartesianCoordinatePlane, 384
  KDChart::PolarCoordinatePlane, 679
Legend
  KDChart::Legend, 500
legend
  KDChart::Chart, 416
  KDChart::Widget, 870
LegendList
  KDChart, 23
legends
  KDChart::Chart, 416
LegendStyle
  KDChart::Legend, 500
legendStyle
  KDChart::Legend, 511
Line
  KDChart::Widget, 866
Linear
  KDChart::AbstractCoordinatePlane, 118
  KDChart::CartesianCoordinatePlane, 368
  KDChart::PolarCoordinatePlane, 669
LineAttributes, 25
  KDChart::LineAttributes, 534
lineAttributes
  KDChart::LineDiagram, 557, 558
LineAttributesRole
  KDChart, 24
LineDiagram
  KDChart::LineDiagram, 542
lineDiagram
  KDChart::Widget, 870
LineLayoutItem
  KDChart::LineLayoutItem, 588
LinesOnly
  KDChart::Legend, 500
LineType
  KDChart::LineDiagram, 541
LineWithMarkerLayoutItem
  KDChart::LineWithMarkerLayoutItem, 593
lineXRotation
  KDChart::ThreeDLineAttributes, 850
lineYRotation
  KDChart::ThreeDLineAttributes, 850
Logarithmic
  KDChart::AbstractCoordinatePlane, 118
  KDChart::CartesianCoordinatePlane, 368
  KDChart::PolarCoordinatePlane, 669

makeDefaultPalette
  KDChartPalette.cpp, 961
makeRainbowPalette
  KDChartPalette.cpp, 961
makeSubduedPalette

KDChartPalette.cpp, 962
mapFromSource
  KDChart::AbstractProxyModel, 247
  KDChart::AttributesModel, 262
  KDChart::DatasetProxyModel, 433
mappingDisabled
  KDChart::DatasetSelectorWidget, 437
mapToSource
  KDChart::AbstractProxyModel, 248
  KDChart::AttributesModel, 262
  KDChart::DatasetProxyModel, 433
Marker1Pixel
  KDChart::MarkerAttributes, 600
Marker4Pixels
  KDChart::MarkerAttributes, 600
MarkerAttributes, 26
  KDChart::MarkerAttributes, 600
markerAttributes
  KDChart::DataValueAttributes, 443
  KDChart::Legend, 512
MarkerCircle
  KDChart::MarkerAttributes, 600
markerColor
  KDChart::MarkerAttributes, 600
MarkerCross
  KDChart::MarkerAttributes, 600
MarkerDiamond
  KDChart::MarkerAttributes, 600
MarkerFastCross
  KDChart::MarkerAttributes, 600
MarkerLayoutItem
  KDChart::MarkerLayoutItem, 603
MarkerRing
  KDChart::MarkerAttributes, 600
MarkersAndLines
  KDChart::Legend, 500
markerSize
  KDChart::MarkerAttributes, 600
MarkersOnly
  KDChart::Legend, 500
MarkerSquare
  KDChart::MarkerAttributes, 600
MarkerStyle
  KDChart::MarkerAttributes, 600
markerStyle
  KDChart::MarkerAttributes, 600
MarkerStylesMap
  KDChart::MarkerAttributes, 599
markerStylesMap
  KDChart::MarkerAttributes, 601
maximumSize
  KDChart::AbstractCoordinatePlane, 126
  KDChart::AutoSpacerLayoutItem, 269
  KDChart::CartesianAxis, 338
  KDChart::CartesianCoordinatePlane, 385
  KDChart::HeaderFooter, 467
  KDChart::HorizontalLineLayoutItem, 480
  KDChart::LineLayoutItem, 588
  KDChart::LineWithMarkerLayoutItem, 594
  KDChart::MarkerLayoutItem, 603
  KDChart::PolarCoordinatePlane, 679
  KDChart::TextArea, 819
  KDChart::TextLayoutItem, 837
  KDChart::VerticalLineLayoutItem, 859
maxPositionValue
  KDChartPosition.cpp, 974
Measure
  KDChart::Measure, 609
MeasureCalculationMode
  KDChartEnums, 487
MeasureCalculationModeAbsolute
  KDChartEnums, 487
MeasureCalculationModeAuto
  KDChartEnums, 487
MeasureCalculationModeAutoArea
  KDChartEnums, 487
MeasureCalculationModeAutoOrientation
  KDChartEnums, 487
MeasureCalculationModeRelative
  KDChartEnums, 487
measureCalculationModeToString
  KDChartEnums, 490
MeasureOrientation
  KDChartEnums, 487
MeasureOrientationAuto
  KDChartEnums, 488
MeasureOrientationHorizontal
  KDChartEnums, 488
MeasureOrientationMaximum
  KDChartEnums, 488
MeasureOrientationMinimum
  KDChartEnums, 488
measureOrientationToString
  KDChartEnums, 491
MeasureOrientationVertical
  KDChartEnums, 488
minimalFontSize
  KDChart::TextAttributes, 831
minimumSize
  KDChart::AbstractCoordinatePlane, 126
  KDChart::AutoSpacerLayoutItem, 269
  KDChart::CartesianAxis, 341
  KDChart::CartesianCoordinatePlane, 385
  KDChart::HeaderFooter, 467
  KDChart::HorizontalLineLayoutItem, 480
  KDChart::LineLayoutItem, 588

KDChart::LineWithMarkerLayoutItem, 594
KDChart::MarkerLayoutItem, 603
KDChart::PolarCoordinatePlane, 679
KDChart::TextArea, 819
KDChart::TextLayoutItem, 838
KDChart::VerticalLineLayoutItem, 859

minimumSizeHint
KDChart::AbstractCoordinatePlane, 126
KDChart::CartesianCoordinatePlane, 385
KDChart::Legend, 512
KDChart::PolarCoordinatePlane, 680
KDTextDocument, 495

MissCount
KDChartTextLabelCache.cpp, 988

MissingValuesAreBridged
KDChart::LineAttributes, 534

MissingValuesHideSegments
KDChart::LineAttributes, 534

MissingValuesPolicy
KDChart::LineAttributes, 533

missingValuesPolicy
KDChart::LineAttributes, 534

MissingValuesPolicyIgnored
KDChart::LineAttributes, 534

MissingValuesShownAsZero
KDChart::LineAttributes, 534

modelData
KDChart::AttributesModel, 262

modelDataMap
KDChart::AttributesModel, 263

modelsChanged
KDChart::AbstractCartesianDiagram, 96
KDChart::AbstractDiagram, 156
KDChart::AbstractPieDiagram, 194
KDChart::AbstractPolarDiagram, 230
KDChart::BarDiagram, 301
KDChart::LineDiagram, 558
KDChart::PieDiagram, 641
KDChart::PolarDiagram, 715
KDChart::RingDiagram, 788

mousePressEvent
KDChart::AbstractCoordinatePlane, 127
KDChart::CartesianCoordinatePlane, 386
KDChart::Chart, 417
KDChart::PolarCoordinatePlane, 680

moveCursor
KDChart::AbstractCartesianDiagram, 96
KDChart::AbstractDiagram, 156
KDChart::AbstractPieDiagram, 194
KDChart::AbstractPolarDiagram, 230
KDChart::BarDiagram, 301
KDChart::LineDiagram, 558
KDChart::PieDiagram, 641
KDChart::PolarDiagram, 715
KDChart::RingDiagram, 788

mParent
KDChart::AbstractArea, 44
KDChart::AbstractAxis, 78
KDChart::AbstractCoordinatePlane, 140
KDChart::AbstractLayoutItem, 175
KDChart::AutoSpacerLayoutItem, 273
KDChart::CartesianAxis, 362
KDChart::CartesianCoordinatePlane, 406
KDChart::HeaderFooter, 477
KDChart::HorizontalLineLayoutItem, 483
KDChart::LineLayoutItem, 592
KDChart::LineWithMarkerLayoutItem, 597
KDChart::MarkerLayoutItem, 607
KDChart::PolarCoordinatePlane, 696
KDChart::TextArea, 828
KDChart::TextLayoutItem, 843
KDChart::VerticalLineLayoutItem, 862

mParentLayout
KDChart::AbstractArea, 44
KDChart::AbstractAxis, 78
KDChart::AbstractCoordinatePlane, 140
KDChart::AbstractLayoutItem, 175
KDChart::AutoSpacerLayoutItem, 273
KDChart::CartesianAxis, 362
KDChart::CartesianCoordinatePlane, 407
KDChart::HeaderFooter, 477
KDChart::HorizontalLineLayoutItem, 483
KDChart::LineLayoutItem, 592
KDChart::LineWithMarkerLayoutItem, 597
KDChart::MarkerLayoutItem, 607
KDChart::PolarCoordinatePlane, 697
KDChart::TextArea, 828
KDChart::TextLayoutItem, 843
KDChart::VerticalLineLayoutItem, 862

mPositionCenter
KDChart::PositionPoints, 747

mPositionEast
KDChart::PositionPoints, 747

mPositionNorth
KDChart::PositionPoints, 747

mPositionNorthEast
KDChart::PositionPoints, 748

mPositionNorthWest
KDChart::PositionPoints, 748

mPositionSouth
KDChart::PositionPoints, 748

mPositionSouthEast
KDChart::PositionPoints, 748

mPositionSouthWest
KDChart::PositionPoints, 748

mPositionUnknown
  KDChart::PositionPoints, 748
mPositionWest
  KDChart::PositionPoints, 748

name
  KDChart::Position, 740
names
  KDChart::Position, 741
needLayoutPlanes
  KDChart::AbstractCoordinatePlane, 127
  KDChart::CartesianCoordinatePlane, 386
  KDChart::PolarCoordinatePlane, 680
needRelayout
  KDChart::AbstractCoordinatePlane, 127
  KDChart::CartesianCoordinatePlane, 386
  KDChart::PolarCoordinatePlane, 680
needSizeHint
  KDChart::AbstractAreaWidget, 54
  KDChart::Legend, 512
needUpdate
  KDChart::AbstractCoordinatePlane, 127
  KDChart::CartesianCoordinatePlane, 386
  KDChart::PolarCoordinatePlane, 680
negativePosition
  KDChart::DataValueAttributes, 443
Normal
  KDChart::BarDiagram, 285
  KDChart::LineDiagram, 541
  KDChart::Widget, 866
North
  KDChart::Position, 743
NorthEast
  KDChart::Position, 743
NorthWest
  KDChart::Position, 743
NoType
  KDChart::Widget, 866
numberOfAbscissaSegments
  KDChart::AbstractCartesianDiagram, 96
  KDChart::BarDiagram, 301
  KDChart::LineDiagram, 558
numberOfGridRings
  KDChart::AbstractPieDiagram, 194
  KDChart::AbstractPolarDiagram, 230
  KDChart::PieDiagram, 641
  KDChart::PolarDiagram, 715
  KDChart::RingDiagram, 789
numberOfOrdinateSegments
  KDChart::AbstractCartesianDiagram, 97
  KDChart::BarDiagram, 301
  KDChart::LineDiagram, 558
numberOfValuesPerDataset
  KDChart::AbstractPieDiagram, 194
  KDChart::AbstractPolarDiagram, 230
  KDChart::PieDiagram, 641
  KDChart::PolarDiagram, 715
  KDChart::RingDiagram, 789

observedBy
  KDChart::AbstractAxis, 69
  KDChart::CartesianAxis, 341
on_cbReverseColumns_stateChanged
  KDChart::DatasetSelectorWidget, 437
on_cbReverseRows_stateChanged
  KDChart::DatasetSelectorWidget, 438
on_groupBox_toggled
  KDChart::DatasetSelectorWidget, 438
on_sbColumnCount_valueChanged
  KDChart::DatasetSelectorWidget, 438
on_sbRowCount_valueChanged
  KDChart::DatasetSelectorWidget, 438
on_sbStartRow_valueChanged
  KDChart::DatasetSelectorWidget, 438
operator!=
  KDChart::AbstractThreeDAttributes, 250
  KDChart::BackgroundAttributes, 275
  KDChart::BarAttributes, 278
  KDChart::DataDimension, 428
  KDChart::DataValueAttributes, 443
  KDChart::FrameAttributes, 452
  KDChart::GridAttributes, 458
  KDChart::LineAttributes, 534
  KDChart::MarkerAttributes, 601
  KDChart::Measure, 611
  KDChart::PieAttributes, 621
  KDChart::Position, 741
  KDChart::RelativePosition, 767
  KDChart::TextAttributes, 831
  KDChart::ThreeDBarAttributes, 845
  KDChart::ThreeDLineAttributes, 851
  KDChart::ThreeDPieAttributes, 855
operator<<
  KDChartAbstractThreeDAttributes.cpp, 904
  KDChartAbstractThreeDAttributes.h, 905
  KDChartBackgroundAttributes.cpp, 908
  KDChartBackgroundAttributes.h, 909
  KDChartDataValueAttributes.cpp, 927
  KDChartDataValueAttributes.h, 929
  KDChartFrameAttributes.cpp, 934
  KDChartFrameAttributes.h, 935
  KDChartGridAttributes.cpp, 941
  KDChartGridAttributes.h, 942
  KDChartLineAttributes.cpp, 951
  KDChartLineAttributes.h, 952
  KDChartMarkerAttributes.cpp, 955
  KDChartMarkerAttributes.h, 956

KDChartMeasure.cpp, 957
KDChartMeasure.h, 958
KDChartPieAttributes.cpp, 964
KDChartPieAttributes.h, 965
KDChartPosition.cpp, 973
KDChartPosition.h, 976
KDChartRelativePosition.cpp, 977
KDChartRelativePosition.h, 978
KDChartTextAttributes.cpp, 985
KDChartTextAttributes.h, 986
KDChartThreeDBarAttributes.cpp, 990
KDChartThreeDBarAttributes.h, 991
KDChartThreeDLineAttributes.cpp, 992
KDChartThreeDLineAttributes.h, 993
KDChartThreeDPieAttributes.cpp, 994
KDChartThreeDPieAttributes.h, 995

operator=
- KDChart::AbstractThreeDAttributes, 250
- KDChart::BackgroundAttributes, 275
- KDChart::BarAttributes, 278
- KDChart::DataValueAttributes, 443
- KDChart::FrameAttributes, 452
- KDChart::GridAttributes, 458
- KDChart::LineAttributes, 534
- KDChart::MarkerAttributes, 601
- KDChart::Measure, 611
- KDChart::Palette, 618
- KDChart::PieAttributes, 621
- KDChart::RelativePosition, 768
- KDChart::TextAttributes, 832
- KDChart::ThreeDBarAttributes, 846
- KDChart::ThreeDLineAttributes, 851
- KDChart::ThreeDPieAttributes, 855

operator==
- KDChart::AbstractThreeDAttributes, 250
- KDChart::BackgroundAttributes, 275
- KDChart::BarAttributes, 278
- KDChart::DataDimension, 428
- KDChart::DataValueAttributes, 443
- KDChart::FrameAttributes, 452
- KDChart::GridAttributes, 459
- KDChart::LineAttributes, 534
- KDChart::MarkerAttributes, 601
- KDChart::Measure, 611
- KDChart::PieAttributes, 621
- KDChart::Position, 741, 742
- KDChart::RelativePosition, 768
- KDChart::TextAttributes, 832
- KDChart::ThreeDBarAttributes, 846
- KDChart::ThreeDLineAttributes, 851
- KDChart::ThreeDPieAttributes, 856

Option
- KDChart::Position, 737

orientation
- KDChart::Legend, 513

padding
- KDChart::FrameAttributes, 452

paint
- KDChart::AbstractArea, 38
- KDChart::AbstractAreaWidget, 55
- KDChart::AbstractAxis, 69
- KDChart::AbstractCartesianDiagram, 97
- KDChart::AbstractCoordinatePlane, 127
- KDChart::AbstractDiagram, 157
- KDChart::AbstractLayoutItem, 173
- KDChart::AbstractPieDiagram, 194
- KDChart::AbstractPolarDiagram, 230
- KDChart::AutoSpacerLayoutItem, 269
- KDChart::BarDiagram, 301
- KDChart::CartesianAxis, 341
- KDChart::CartesianCoordinatePlane, 386
- KDChart::Chart, 417
- KDChart::HeaderFooter, 467
- KDChart::HorizontalLineLayoutItem, 480
- KDChart::Legend, 513
- KDChart::LineDiagram, 559
- KDChart::LineLayoutItem, 589
- KDChart::LineWithMarkerLayoutItem, 595
- KDChart::MarkerLayoutItem, 604
- KDChart::PieDiagram, 641
- KDChart::PolarCoordinatePlane, 680
- KDChart::PolarDiagram, 716
- KDChart::RingDiagram, 789
- KDChart::TextArea, 819
- KDChart::TextLayoutItem, 838
- KDChart::VerticalLineLayoutItem, 859

paintAll
- KDChart::AbstractArea, 38
- KDChart::AbstractAreaWidget, 55
- KDChart::AbstractAxis, 69
- KDChart::AbstractCoordinatePlane, 127
- KDChart::AbstractLayoutItem, 174
- KDChart::AutoSpacerLayoutItem, 270
- KDChart::CartesianAxis, 342
- KDChart::CartesianCoordinatePlane, 387
- KDChart::HeaderFooter, 468
- KDChart::HorizontalLineLayoutItem, 481
- KDChart::Legend, 514
- KDChart::LineLayoutItem, 589
- KDChart::LineWithMarkerLayoutItem, 595
- KDChart::MarkerLayoutItem, 604
- KDChart::PolarCoordinatePlane, 681
- KDChart::TextArea, 820
- KDChart::TextLayoutItem, 839
- KDChart::VerticalLineLayoutItem, 860

paintBackground
  KDChart::AbstractArea, 39
  KDChart::AbstractAreaBase, 48
  KDChart::AbstractAreaWidget, 56
  KDChart::AbstractAxis, 70
  KDChart::AbstractCoordinatePlane, 128
  KDChart::CartesianAxis, 343
  KDChart::CartesianCoordinatePlane, 388
  KDChart::HeaderFooter, 469
  KDChart::Legend, 514
  KDChart::PolarCoordinatePlane, 682
  KDChart::TextArea, 821
paintBackgroundAttributes
  KDChart::AbstractArea, 39
  KDChart::AbstractAreaBase, 48
  KDChart::AbstractAreaWidget, 56
  KDChart::AbstractAxis, 70
  KDChart::AbstractCoordinatePlane, 128
  KDChart::CartesianAxis, 343
  KDChart::CartesianCoordinatePlane, 388
  KDChart::HeaderFooter, 469
  KDChart::Legend, 515
  KDChart::PolarCoordinatePlane, 682
  KDChart::TextArea, 821
PaintContext, 27
  KDChart::PaintContext, 615
paintCtx
  KDChart::AbstractArea, 40
  KDChart::AbstractAxis, 71
  KDChart::AbstractCoordinatePlane, 129
  KDChart::AbstractLayoutItem, 174
  KDChart::AutoSpacerLayoutItem, 271
  KDChart::CartesianAxis, 344
  KDChart::CartesianCoordinatePlane, 389
  KDChart::HeaderFooter, 470
  KDChart::HorizontalLineLayoutItem, 481
  KDChart::LineLayoutItem, 589
  KDChart::LineWithMarkerLayoutItem, 595
  KDChart::MarkerLayoutItem, 604
  KDChart::PolarCoordinatePlane, 683
  KDChart::TextArea, 822
  KDChart::TextLayoutItem, 839
  KDChart::VerticalLineLayoutItem, 860
paintDataValueText
  KDChart::AbstractCartesianDiagram, 97
  KDChart::AbstractDiagram, 157
  KDChart::AbstractPieDiagram, 194
  KDChart::AbstractPolarDiagram, 231
  KDChart::BarDiagram, 305
  KDChart::LineDiagram, 564
  KDChart::PieDiagram, 644
  KDChart::PolarDiagram, 716
  KDChart::RingDiagram, 790
paintDataValueTexts
  KDChart::AbstractCartesianDiagram, 98
  KDChart::AbstractDiagram, 158
  KDChart::AbstractPieDiagram, 196
  KDChart::AbstractPolarDiagram, 232
  KDChart::BarDiagram, 306
  KDChart::LineDiagram, 566
  KDChart::PieDiagram, 646
  KDChart::PolarDiagram, 718
  KDChart::RingDiagram, 791
painter
  KDChart::PaintContext, 615
paintEvent
  KDChart::AbstractAreaWidget, 57
  KDChart::CartesianCoordinatePlane, 389
  KDChart::Chart, 418
  KDChart::Legend, 516
  KDChart::LineDiagram, 566
  KDChart::PieDiagram, 646
  KDChart::PolarCoordinatePlane, 683
  KDChart::PolarDiagram, 718
  KDChart::RingDiagram, 791
paintFrame
  KDChart::AbstractArea, 40
  KDChart::AbstractAreaBase, 49
  KDChart::AbstractAreaWidget, 57
  KDChart::AbstractAxis, 72
  KDChart::AbstractCoordinatePlane, 130
  KDChart::CartesianAxis, 353
  KDChart::CartesianCoordinatePlane, 390
  KDChart::HeaderFooter, 470
  KDChart::Legend, 516
  KDChart::PolarCoordinatePlane, 683
  KDChart::TextArea, 822
paintFrameAttributes
  KDChart::AbstractArea, 40
  KDChart::AbstractAreaBase, 49
  KDChart::AbstractAreaWidget, 58
  KDChart::AbstractAxis, 72
  KDChart::AbstractCoordinatePlane, 130
  KDChart::CartesianAxis, 353
  KDChart::CartesianCoordinatePlane, 390
  KDChart::HeaderFooter, 471
  KDChart::Legend, 516
  KDChart::PolarCoordinatePlane, 683
  KDChart::TextArea, 823
paintIntoRect
  KDChart::AbstractArea, 41
  KDChart::AbstractAreaWidget, 58
  KDChart::AbstractAxis, 72
  KDChart::AbstractCoordinatePlane, 130
  KDChart::CartesianAxis, 354
  KDChart::CartesianCoordinatePlane, 390
  KDChart::HeaderFooter, 471

  KDChart::Legend, 517
  KDChart::LineLayoutItem, 589
  KDChart::MarkerLayoutItem, 604
  KDChart::PolarCoordinatePlane, 684
  KDChart::TextArea, 823
paintMarker
  KDChart::AbstractCartesianDiagram, 99
  KDChart::AbstractDiagram, 159
  KDChart::AbstractPieDiagram, 196, 197
  KDChart::AbstractPolarDiagram, 232, 233
  KDChart::BarDiagram, 307
  KDChart::LineDiagram, 566, 567
  KDChart::PieDiagram, 646, 647
  KDChart::PolarDiagram, 718, 719
  KDChart::RingDiagram, 792
paintMarkers
  KDChart::AbstractCartesianDiagram, 101
  KDChart::AbstractDiagram, 161
  KDChart::AbstractPieDiagram, 198
  KDChart::AbstractPolarDiagram, 234
  KDChart::BarDiagram, 309
  KDChart::LineDiagram, 569
  KDChart::PieDiagram, 648
  KDChart::PolarDiagram, 721
  KDChart::RingDiagram, 794
paintPolarMarkers
  KDChart::PolarDiagram, 721
Palette, 28
  KDChart::Palette, 618
PaletteType
  KDChart::AttributesModel, 254
paletteType
  KDChart::AttributesModel, 263
PaletteTypeDefault
  KDChart::AttributesModel, 254
PaletteTypeRainbow
  KDChart::AttributesModel, 254
PaletteTypeSubdued
  KDChart::AttributesModel, 254
parent
  KDChart::AbstractCoordinatePlane, 131
  KDChart::AbstractProxyModel, 248
  KDChart::AttributesModel, 263
  KDChart::CartesianCoordinatePlane, 391
  KDChart::DatasetProxyModel, 434
  KDChart::PolarCoordinatePlane, 684
parentLayout
  KDChart::AbstractArea, 41
  KDChart::AbstractAxis, 73
  KDChart::AbstractCoordinatePlane, 131
  KDChart::AbstractLayoutItem, 174
  KDChart::AutoSpacerLayoutItem, 271
  KDChart::CartesianAxis, 354
  KDChart::CartesianCoordinatePlane, 391
  KDChart::HeaderFooter, 471
  KDChart::HorizontalLineLayoutItem, 481
  KDChart::LineLayoutItem, 590
  KDChart::LineWithMarkerLayoutItem, 596
  KDChart::MarkerLayoutItem, 605
  KDChart::PolarCoordinatePlane, 685
  KDChart::TextArea, 823
  KDChart::TextLayoutItem, 839
  KDChart::VerticalLineLayoutItem, 860
pen
  KDChart::AbstractCartesianDiagram, 101, 102
  KDChart::AbstractDiagram, 161, 162
  KDChart::AbstractPieDiagram, 199
  KDChart::AbstractPolarDiagram, 235
  KDChart::BarDiagram, 309, 310
  KDChart::FrameAttributes, 452
  KDChart::Legend, 517
  KDChart::LineDiagram, 569, 570
  KDChart::MarkerAttributes, 601
  KDChart::PieDiagram, 649
  KDChart::PolarDiagram, 721, 722
  KDChart::RingDiagram, 794, 795
  KDChart::TextAttributes, 832
  PrerenderedLabel, 754
pens
  KDChart::Legend, 518
Percent
  KDChart::BarDiagram, 285
  KDChart::LineDiagram, 541
  KDChart::Widget, 866
percentMode
  KDChart::AbstractCartesianDiagram, 102
  KDChart::AbstractDiagram, 162
  KDChart::AbstractPieDiagram, 199
  KDChart::AbstractPolarDiagram, 236
  KDChart::BarDiagram, 310
  KDChart::LineDiagram, 570
  KDChart::PieDiagram, 650
  KDChart::PolarDiagram, 722
  KDChart::RingDiagram, 795
PI
  KDChartLayoutItems.cpp, 945
Pie
  KDChart::Widget, 866
PieAttributes
  KDChart::PieAttributes, 620
pieAttributes
  KDChart::AbstractPieDiagram, 200
  KDChart::PieDiagram, 650
  KDChart::RingDiagram, 795, 796
PieAttributesRole
  KDChart, 24

PieDiagram
  KDChart::PieDiagram, 628
pieDiagram
  KDChart::Widget, 871
pixmap
  KDChart::BackgroundAttributes, 275
  PrerenderedElement, 750
  PrerenderedLabel, 754
pixmapMode
  KDChart::BackgroundAttributes, 275
point
  KDChart::PositionPoints, 747
Polar
  KDChart::Widget, 866
PolarCoordinatePlane
  KDChart::PolarCoordinatePlane, 669
polarCoordinatePlane
  KDChart::AbstractPieDiagram, 200
  KDChart::AbstractPolarDiagram, 236
  KDChart::PieDiagram, 651
  KDChart::PolarDiagram, 722
  KDChart::RingDiagram, 796
PolarDiagram
  KDChart::PolarDiagram, 703
polarDiagram
  KDChart::Widget, 871
Position
  KDChart::CartesianAxis, 329
  KDChart::Position, 738
position
  KDChart::CartesianAxis, 354
  KDChart::DataValueAttributes, 443
  KDChart::HeaderFooter, 472
  KDChart::Legend, 518
  PrerenderedElement, 750
  PrerenderedLabel, 754
PositionCenter
  KDChartEnums, 488
positionChanged
  KDChart::HeaderFooter, 472
PositionEast
  KDChartEnums, 489
PositionFloating
  KDChartEnums, 489
positionHasChanged
  KDChart::AbstractArea, 41
  KDChart::AbstractAreaBase, 50
  KDChart::AbstractAreaWidget, 59
  KDChart::AbstractAxis, 73
  KDChart::AbstractCoordinatePlane, 131
  KDChart::CartesianAxis, 354
  KDChart::CartesianCoordinatePlane, 391
  KDChart::HeaderFooter, 472
  KDChart::Legend, 518
  KDChart::PolarCoordinatePlane, 685
  KDChart::TextArea, 824
PositionNorth
  KDChartEnums, 489
PositionNorthEast
  KDChartEnums, 489
PositionNorthWest
  KDChartEnums, 489
PositionPoints
  KDChart::PositionPoints, 745, 746
PositionSouth
  KDChartEnums, 489
PositionSouthEast
  KDChartEnums, 489
PositionSouthWest
  KDChartEnums, 489
PositionUnknown
  KDChartEnums, 488
PositionValue
  KDChartEnums, 488
PositionWest
  KDChartEnums, 489
positivePosition
  KDChart::DataValueAttributes, 443
powerOfTenDivisor
  KDChart::DataValueAttributes, 444
prefix
  KDChart::DataValueAttributes, 444
PrerenderedElement, 749
  PrerenderedElement, 749
PrerenderedElement
  ~PrerenderedElement, 749
  invalidate, 750
  pixmap, 750
  position, 750
  PrerenderedElement, 749
  referencePoint, 750
  referencePointLocation, 750
  setPosition, 750
  setReferencePoint, 751
PrerenderedLabel, 752
  PrerenderedLabel, 753
PrerenderedLabel
  ~PrerenderedLabel, 753
  angle, 753
  brush, 753
  font, 754
  invalidate, 754
  pen, 754
  pixmap, 754
  position, 754
  PrerenderedLabel, 753
  referencePoint, 755
  referencePointLocation, 755

  setAngle, 756
  setBrush, 756
  setFont, 756
  setPen, 756
  setPosition, 757
  setReferencePoint, 757
  setText, 757
  text, 757
printableName
  KDChart::Position, 742
printableNames
  KDChart::Position, 742
propertiesChanged
  KDChart::AbstractCartesianDiagram, 102
  KDChart::AbstractCoordinatePlane, 131
  KDChart::AbstractDiagram, 162
  KDChart::AbstractPieDiagram, 201
  KDChart::AbstractPolarDiagram, 236
  KDChart::BarDiagram, 310
  KDChart::CartesianCoordinatePlane, 391
  KDChart::Legend, 518
  KDChart::LineDiagram, 570
  KDChart::PieDiagram, 651
  KDChart::PolarCoordinatePlane, 685
  KDChart::PolarDiagram, 722
  KDChart::RingDiagram, 796

Q_DECLARE_TYPEINFO
  KDChartBackgroundAttributes.h, 909
  KDChartDataValueAttributes.h, 930
  KDChartFrameAttributes.h, 935
  KDChartGridAttributes.h, 942
  KDChartLineAttributes.h, 952
  KDChartMarkerAttributes.h, 956
  KDChartPieAttributes.h, 965
  KDChartPosition.h, 976
  KDChartRelativePosition.h, 978
  KDChartTextAttributes.h, 986
  KDChartThreeDBarAttributes.h, 991
  KDChartThreeDLineAttributes.h, 993
  KDChartThreeDPieAttributes.h, 995
Q_SLOTS
  KDChart::AbstractAxis, 78
  KDChart::AbstractCoordinatePlane, 140
  KDChart::CartesianAxis, 362
  KDChart::CartesianCoordinatePlane, 407
  KDChart::DatasetProxyModel, 436
  KDChart::DatasetSelectorWidget, 439
  KDChart::DiagramObserver, 451
  KDChart::Legend, 532
  KDChart::PolarCoordinatePlane, 697
  KDChart::SignalCompressor, 811, 812
  KDChart::Widget, 877
QAbstractItemView, 758
QAbstractProxyModel, 759
QFrame, 760
QLayoutItem, 761
QObject, 762
QSortFilterProxyModel, 763
QTextDocument, 764
QWidget, 765

radiusUnit
  KDChart::PolarCoordinatePlane, 685
rainbowPalette
  KDChart::Palette, 618
realFont
  KDChart::HeaderFooter, 472
  KDChart::TextArea, 824
  KDChart::TextLayoutItem, 839
realFontSize
  KDChart::HeaderFooter, 472
  KDChart::TextArea, 824
  KDChart::TextLayoutItem, 839
rectangle
  KDChart::PaintContext, 615
referenceArea
  KDChart::Legend, 519
  KDChart::Measure, 611
  KDChart::RelativePosition, 768
referenceCoordinatePlane
  KDChart::AbstractCoordinatePlane, 132
  KDChart::CartesianCoordinatePlane, 392
  KDChart::PolarCoordinatePlane, 686
referenceDiagram
  KDChart::AbstractCartesianDiagram, 102
  KDChart::BarDiagram, 310
  KDChart::LineDiagram, 570
referenceDiagramIsBarDiagram
  KDChartCartesianAxis.cpp, 915
referenceDiagramOffset
  KDChart::AbstractCartesianDiagram, 102
  KDChart::BarDiagram, 311
  KDChart::LineDiagram, 570
referenceOrientation
  KDChart::Measure, 611
referencePoint
  KDChart::RelativePosition, 768
  PrerenderedElement, 750
  PrerenderedLabel, 755
referencePointLocation
  PrerenderedElement, 750
  PrerenderedLabel, 755
referencePoints
  KDChart::RelativePosition, 768
referencePosition
  KDChart::RelativePosition, 768
RelativePosition, 29

KDChart::RelativePosition, 767
relativeThickness
  KDChart::RingDiagram, 796
relayout
  KDChart::AbstractCoordinatePlane, 132
  KDChart::CartesianCoordinatePlane, 392
  KDChart::PolarCoordinatePlane, 686
reLayoutFloatingLegends
  KDChart::Chart, 419
removeBrush
  KDChart::Palette, 618
removeDiagram
  KDChart::Legend, 519
removeDiagrams
  KDChart::Legend, 519
removeFromParentLayout
  KDChart::AbstractArea, 41
  KDChart::AbstractAxis, 73
  KDChart::AbstractCoordinatePlane, 132
  KDChart::AbstractLayoutItem, 174
  KDChart::AutoSpacerLayoutItem, 271
  KDChart::CartesianAxis, 355
  KDChart::CartesianCoordinatePlane, 392
  KDChart::HeaderFooter, 472
  KDChart::HorizontalLineLayoutItem, 481
  KDChart::LineLayoutItem, 590
  KDChart::LineWithMarkerLayoutItem, 596
  KDChart::MarkerLayoutItem, 605
  KDChart::PolarCoordinatePlane, 686
  KDChart::TextArea, 824
  KDChart::TextLayoutItem, 840
  KDChart::VerticalLineLayoutItem, 860
replaceCoordinatePlane
  KDChart::Chart, 419
replaceDiagram
  KDChart::AbstractCoordinatePlane, 133
  KDChart::CartesianCoordinatePlane, 393
  KDChart::Legend, 520
  KDChart::PolarCoordinatePlane, 687
replaceHeaderFooter
  KDChart::Chart, 420
  KDChart::Widget, 871
replaceLegend
  KDChart::Chart, 420
  KDChart::Widget, 872
resetData
  KDChart::AttributesModel, 263
  KDChart::Widget, 872
resetDiagram
  KDChart::Legend, 520
resetFactors
  KDChart::GlobalMeasureScaling, 455
resetGridAttributes
  KDChart::CartesianCoordinatePlane, 393
  KDChart::PolarCoordinatePlane, 687
resetHeaderData
  KDChart::AttributesModel, 263
resetLineAttributes
  KDChart::LineDiagram, 571
resetReferencePosition
  KDChart::RelativePosition, 768
resetTexts
  KDChart::Legend, 520
resetTitleTextAttributes
  KDChart::CartesianAxis, 355
resize
  KDChart::AbstractCartesianDiagram, 103
  KDChart::AbstractDiagram, 162
  KDChart::AbstractPieDiagram, 201
  KDChart::AbstractPolarDiagram, 236
  KDChart::BarDiagram, 311
  KDChart::LineDiagram, 571
  KDChart::PieDiagram, 651
  KDChart::PolarDiagram, 723
  KDChart::RingDiagram, 796
resizeEvent
  KDChart::BarDiagram, 311
  KDChart::Chart, 421
  KDChart::Legend, 521
  KDChart::LineDiagram, 571
  KDChart::PieDiagram, 651
  KDChart::PolarCoordinatePlane, 688
  KDChart::PolarDiagram, 723
  KDChart::RingDiagram, 797
resizeLayout
  KDChart::AbstractAreaWidget, 59
  KDChart::Legend, 521
Right
  KDChart::CartesianAxis, 329
right
  KDChart::Widget, 877
rightOverlap
  KDChart::AbstractArea, 42
  KDChart::AbstractAxis, 73
  KDChart::AbstractCoordinatePlane, 133
  KDChart::CartesianAxis, 355
  KDChart::CartesianCoordinatePlane, 394
  KDChart::PolarCoordinatePlane, 688
Ring
  KDChart::Widget, 866
RingDiagram
  KDChart::RingDiagram, 776
ringDiagram
  KDChart::Widget, 872
rotateCircularLabels
  KDChart::PolarDiagram, 723
rotatedPoint

KDChartLayoutItems.cpp, 946
rotatedRect
  KDChartLayoutItems.cpp, 946
rotation
  KDChart::RelativePosition, 768
  KDChart::TextAttributes, 832
rowCount
  KDChart::AttributesModel, 264
Rows
  KDChart::BarDiagram, 285
  KDChart::Widget, 866

scrollTo
  KDChart::AbstractCartesianDiagram, 103
  KDChart::AbstractDiagram, 162
  KDChart::AbstractPieDiagram, 201
  KDChart::AbstractPolarDiagram, 236
  KDChart::BarDiagram, 311
  KDChart::LineDiagram, 571
  KDChart::PieDiagram, 651
  KDChart::PolarDiagram, 723
  KDChart::RingDiagram, 797
sequence
  KDChart::DataDimension, 429
SET_ALL_MARGINS_TO_ZERO
  KDChartChart.cpp, 920
SET_SUB_TYPE
  KDChartWidget.cpp, 996
setAbsoluteValue
  KDChart::Measure, 612
setAdjustBoundsToGrid
  KDChart::GridAttributes, 459
setAlignment
  KDChart::Legend, 521
  KDChart::RelativePosition, 769
setAllowOverlappingDataValueTexts
  KDChart::AbstractCartesianDiagram, 103
  KDChart::AbstractDiagram, 163
  KDChart::AbstractPieDiagram, 201
  KDChart::AbstractPolarDiagram, 237
  KDChart::BarDiagram, 312
  KDChart::LineDiagram, 572
  KDChart::PieDiagram, 651
  KDChart::PolarDiagram, 723
  KDChart::RingDiagram, 797
setAngle
  KDChart::ThreeDBarAttributes, 846
  PrerenderedLabel, 756
setAntiAliasing
  KDChart::AbstractCartesianDiagram, 103
  KDChart::AbstractDiagram, 163
  KDChart::AbstractPieDiagram, 201
  KDChart::AbstractPolarDiagram, 237
  KDChart::BarDiagram, 312
  KDChart::LineDiagram, 572
  KDChart::PieDiagram, 652
  KDChart::PolarDiagram, 724
  KDChart::RingDiagram, 797
setAttributesModel
  KDChart::AbstractCartesianDiagram, 104
  KDChart::AbstractDiagram, 163
  KDChart::AbstractPieDiagram, 202
  KDChart::AbstractPolarDiagram, 237
  KDChart::BarDiagram, 312
  KDChart::LineDiagram, 572
  KDChart::PieDiagram, 652
  KDChart::PolarDiagram, 724
  KDChart::RingDiagram, 797
setAttributesModelRootIndex
  KDChart::AbstractCartesianDiagram, 104
  KDChart::AbstractDiagram, 164
  KDChart::AbstractPieDiagram, 202
  KDChart::AbstractPolarDiagram, 238
  KDChart::BarDiagram, 313
  KDChart::LineDiagram, 573
  KDChart::PieDiagram, 653
  KDChart::PolarDiagram, 725
  KDChart::RingDiagram, 798
setAutoAdjustGridToZoom
  KDChart::CartesianCoordinatePlane, 394
setAutoAdjustHorizontalRangeToData
  KDChart::CartesianCoordinatePlane, 395
setAutoAdjustVerticalRangeToData
  KDChart::CartesianCoordinatePlane, 395
setAutoReferenceArea
  KDChart::HeaderFooter, 473
  KDChart::TextArea, 824
  KDChart::TextLayoutItem, 840
setAutoRotate
  KDChart::TextAttributes, 832
setAutoShrink
  KDChart::TextAttributes, 832
setAxesCalcModes
  KDChart::CartesianCoordinatePlane, 396
setAxesCalcModeX
  KDChart::CartesianCoordinatePlane, 396
setAxesCalcModeY
  KDChart::CartesianCoordinatePlane, 396
setBackgroundAttributes
  KDChart::AbstractArea, 42
  KDChart::AbstractAreaBase, 50
  KDChart::AbstractAreaWidget, 59
  KDChart::AbstractAxis, 74
  KDChart::AbstractCoordinatePlane, 134
  KDChart::CartesianAxis, 356
  KDChart::CartesianCoordinatePlane, 397
  KDChart::Chart, 421
  KDChart::DataValueAttributes, 444

KDChart::HeaderFooter, 473
KDChart::Legend, 522
KDChart::PolarCoordinatePlane, 688
KDChart::TextArea, 825
setBarAttributes
KDChart::BarDiagram, 313, 314
setBarGapFactor
KDChart::BarAttributes, 278
setBrush
KDChart::AbstractCartesianDiagram, 105
KDChart::AbstractDiagram, 164, 165
KDChart::AbstractPieDiagram, 203
KDChart::AbstractPolarDiagram, 238, 239
KDChart::BackgroundAttributes, 275
KDChart::BarDiagram, 314
KDChart::Legend, 522
KDChart::LineDiagram, 573, 574
KDChart::PieDiagram, 653
KDChart::PolarDiagram, 725
KDChart::RingDiagram, 798, 799
PrerenderedLabel, 756
setBrushesFromDiagram
KDChart::Legend, 522
setCalculationMode
KDChart::Measure, 612
setCenter
KDChart::ZoomParameters, 879
setCloseDatasets
KDChart::PolarDiagram, 726
setColor
KDChart::Legend, 522
setCoordinatePlane
KDChart::AbstractCartesianDiagram, 105
KDChart::AbstractDiagram, 165
KDChart::AbstractPieDiagram, 203
KDChart::AbstractPolarDiagram, 239
KDChart::BarDiagram, 314
KDChart::LineDiagram, 574
KDChart::PaintContext, 615
KDChart::PieDiagram, 654
KDChart::PolarDiagram, 726
KDChart::RingDiagram, 799
setCoordinatePlaneLayout
KDChart::Chart, 422
setData
KDChart::AttributesModel, 264
setDataBoundariesDirty
KDChart::AbstractCartesianDiagram, 106
KDChart::AbstractDiagram, 165
KDChart::AbstractPieDiagram, 204
KDChart::AbstractPolarDiagram, 239
KDChart::BarDiagram, 315
KDChart::LineDiagram, 574
KDChart::PieDiagram, 654
KDChart::PolarDiagram, 726
KDChart::RingDiagram, 799
setDataCell
KDChart::Widget, 872, 873
setDataLabel
KDChart::DataValueAttributes, 444
setDataMap
KDChart::AttributesModel, 264
setDataset
KDChart::Widget, 873
setDatasetColumnDescriptionVector
KDChart::DatasetProxyModel, 434
setDatasetDescriptionVectors
KDChart::DatasetProxyModel, 434
setDatasetDimension
KDChart::AbstractCartesianDiagram, 106
KDChart::AbstractDiagram, 165
KDChart::AbstractPieDiagram, 204
KDChart::AbstractPolarDiagram, 239
KDChart::BarDiagram, 315
KDChart::LineDiagram, 575
KDChart::PieDiagram, 654
KDChart::PolarDiagram, 727
KDChart::RingDiagram, 800
setDatasetRowDescriptionVector
KDChart::DatasetProxyModel, 435
setDataValueAttributes
KDChart::AbstractCartesianDiagram, 107
KDChart::AbstractDiagram, 166
KDChart::AbstractPieDiagram, 204, 205
KDChart::AbstractPolarDiagram, 240
KDChart::BarDiagram, 316
KDChart::LineDiagram, 575, 576
KDChart::PieDiagram, 655
KDChart::PolarDiagram, 727
KDChart::RingDiagram, 800, 801
setDecimalDigits
KDChart::DataValueAttributes, 444
setDefaultColors
KDChart::Legend, 523
setDepth
KDChart::AbstractThreeDAttributes, 251
KDChart::ThreeDBarAttributes, 847
KDChart::ThreeDLineAttributes, 852
KDChart::ThreeDPieAttributes, 856
setDiagram
KDChart::Legend, 523
setDisplayArea
KDChart::LineAttributes, 534
setDrawSolidExcessArrows
KDChart::BarAttributes, 278
setEnabled
KDChart::AbstractThreeDAttributes, 251
KDChart::ThreeDBarAttributes, 847

KDChart::ThreeDLineAttributes, 852
KDChart::ThreeDPieAttributes, 856
setExplode
KDChart::PieAttributes, 622
setExplodeFactor
KDChart::PieAttributes, 622
setFactors
KDChart::GlobalMeasureScaling, 455
setFixedBarWidth
KDChart::BarAttributes, 278
setFixedDataValueGap
KDChart::BarAttributes, 278
setFixedValueBlockGap
KDChart::BarAttributes, 278
setFloatingPosition
KDChart::Legend, 523
setFont
KDChart::TextAttributes, 832
PrerenderedLabel, 756
setFontSize
KDChart::TextAttributes, 832
setFrameAttributes
KDChart::AbstractArea, 42
KDChart::AbstractAreaBase, 50
KDChart::AbstractAreaWidget, 59
KDChart::AbstractAxis, 74
KDChart::AbstractCoordinatePlane, 134
KDChart::CartesianAxis, 356
KDChart::CartesianCoordinatePlane, 397
KDChart::Chart, 422
KDChart::DataValueAttributes, 444
KDChart::HeaderFooter, 473
KDChart::Legend, 524
KDChart::PolarCoordinatePlane, 688
KDChart::TextArea, 825
setGeometry
KDChart::AbstractAxis, 74
KDChart::AbstractCoordinatePlane, 134
KDChart::AutoSpacerLayoutItem, 271
KDChart::CartesianAxis, 356
KDChart::CartesianCoordinatePlane, 397
KDChart::HeaderFooter, 473
KDChart::HorizontalLineLayoutItem, 482
KDChart::LineLayoutItem, 590
KDChart::LineWithMarkerLayoutItem, 596
KDChart::MarkerLayoutItem, 606
KDChart::PolarCoordinatePlane, 689
KDChart::TextArea, 825
KDChart::TextLayoutItem, 840
KDChart::VerticalLineLayoutItem, 861
setGlobalGridAttributes
KDChart::AbstractCoordinatePlane, 135
KDChart::CartesianCoordinatePlane, 398
KDChart::PolarCoordinatePlane, 689
setGlobalLeading
KDChart::Chart, 422
setGlobalLeadingBottom
KDChart::Chart, 423
KDChart::Widget, 874
setGlobalLeadingLeft
KDChart::Chart, 423
KDChart::Widget, 874
setGlobalLeadingRight
KDChart::Chart, 424
KDChart::Widget, 874
setGlobalLeadingTop
KDChart::Chart, 424
KDChart::Widget, 875
setGranularity
KDChart::AbstractPieDiagram, 205
KDChart::PieDiagram, 656
KDChart::RingDiagram, 801
setGridAttributes
KDChart::CartesianCoordinatePlane, 398
KDChart::PolarCoordinatePlane, 689
setGridGranularitySequence
KDChart::GridAttributes, 459
setGridNeedsRecalculate
KDChart::AbstractCoordinatePlane, 135
KDChart::CartesianCoordinatePlane, 399
KDChart::PolarCoordinatePlane, 690
setGridPen
KDChart::GridAttributes, 459
setGridStepWidth
KDChart::GridAttributes, 459
setGridSubStepWidth
KDChart::GridAttributes, 459
setGridVisible
KDChart::GridAttributes, 459
setGroupGapFactor
KDChart::BarAttributes, 278
setHeaderData
KDChart::AttributesModel, 265
setHidden
KDChart::AbstractCartesianDiagram, 108
KDChart::AbstractDiagram, 167
KDChart::AbstractPieDiagram, 206
KDChart::AbstractPolarDiagram, 241, 242
KDChart::BarDiagram, 317
KDChart::LineDiagram, 576, 577
KDChart::PieDiagram, 656, 657
KDChart::PolarDiagram, 728, 729
KDChart::RingDiagram, 801, 802
setHorizontalHeaderDataMap
KDChart::AttributesModel, 265
setHorizontalPadding
KDChart::RelativePosition, 769

setHorizontalRange
    KDChart::CartesianCoordinatePlane, 399
setIsometricScaling
    KDChart::CartesianCoordinatePlane, 400
setLabels
    KDChart::AbstractAxis, 74
    KDChart::CartesianAxis, 356
setLegendStyle
    KDChart::Legend, 525
setLineAttributes
    KDChart::LineDiagram, 577, 578
setLineXRotation
    KDChart::ThreeDLineAttributes, 852
setLineYRotation
    KDChart::ThreeDLineAttributes, 852
setMarkerAttributes
    KDChart::DataValueAttributes, 445
    KDChart::Legend, 525
setMarkerColor
    KDChart::MarkerAttributes, 601
setMarkerSize
    KDChart::MarkerAttributes, 601
setMarkerStyle
    KDChart::MarkerAttributes, 601
setMarkerStylesMap
    KDChart::MarkerAttributes, 601
setMinimalFontSize
    KDChart::TextAttributes, 833
setMissingValuesPolicy
    KDChart::LineAttributes, 534
setModel
    KDChart::AbstractCartesianDiagram, 109
    KDChart::AbstractDiagram, 168
    KDChart::AbstractPieDiagram, 207
    KDChart::AbstractPolarDiagram, 242
    KDChart::BarDiagram, 318
    KDChart::LineDiagram, 578
    KDChart::PieDiagram, 657
    KDChart::PolarDiagram, 729
    KDChart::RingDiagram, 803
setModelData
    KDChart::AttributesModel, 265
setModelDataMap
    KDChart::AttributesModel, 265
setNeedRebuild
    KDChart::Legend, 525
setNegativePosition
    KDChart::DataValueAttributes, 445
setOrientation
    KDChart::Legend, 525
setPadding
    KDChart::FrameAttributes, 453
setPainter
    KDChart::PaintContext, 615
setPaletteType
    KDChart::AttributesModel, 266
setParent
    KDChart::AbstractCoordinatePlane, 135
    KDChart::CartesianCoordinatePlane, 400
    KDChart::HeaderFooter, 474
    KDChart::PolarCoordinatePlane, 690
setParentLayout
    KDChart::AbstractArea, 43
    KDChart::AbstractAxis, 75
    KDChart::AbstractCoordinatePlane, 136
    KDChart::AbstractLayoutItem, 174
    KDChart::AutoSpacerLayoutItem, 271
    KDChart::CartesianAxis, 357
    KDChart::CartesianCoordinatePlane, 400
    KDChart::HeaderFooter, 474
    KDChart::HorizontalLineLayoutItem, 482
    KDChart::LineLayoutItem, 590
    KDChart::LineWithMarkerLayoutItem, 596
    KDChart::MarkerLayoutItem, 606
    KDChart::PolarCoordinatePlane, 691
    KDChart::TextArea, 825
    KDChart::TextLayoutItem, 840
    KDChart::VerticalLineLayoutItem, 861
setParentWidget
    KDChart::AbstractArea, 43
    KDChart::AbstractAxis, 75
    KDChart::AbstractCoordinatePlane, 136
    KDChart::AbstractLayoutItem, 175
    KDChart::AutoSpacerLayoutItem, 272
    KDChart::CartesianAxis, 357
    KDChart::CartesianCoordinatePlane, 400
    KDChart::HeaderFooter, 474
    KDChart::HorizontalLineLayoutItem, 482
    KDChart::LineLayoutItem, 591
    KDChart::LineWithMarkerLayoutItem, 596
    KDChart::MarkerLayoutItem, 606
    KDChart::PolarCoordinatePlane, 691
    KDChart::TextArea, 826
    KDChart::TextLayoutItem, 841
    KDChart::VerticalLineLayoutItem, 861
setPen
    KDChart::AbstractCartesianDiagram, 109, 110
    KDChart::AbstractDiagram, 168, 169
    KDChart::AbstractPieDiagram, 207, 208
    KDChart::AbstractPolarDiagram, 242, 243
    KDChart::BarDiagram, 318, 319
    KDChart::FrameAttributes, 453
    KDChart::Legend, 526
    KDChart::LineDiagram, 578, 579
    KDChart::MarkerAttributes, 601

    KDChart::PieDiagram, 658
    KDChart::PolarDiagram, 729, 730
    KDChart::RingDiagram, 803
    KDChart::TextAttributes, 833
    PrerenderedLabel, 756
setPercentMode
    KDChart::AbstractCartesianDiagram, 110
    KDChart::AbstractDiagram, 169
    KDChart::AbstractPieDiagram, 208
    KDChart::AbstractPolarDiagram, 243
    KDChart::BarDiagram, 319
    KDChart::LineDiagram, 579
    KDChart::PieDiagram, 658
    KDChart::PolarDiagram, 730
    KDChart::RingDiagram, 804
setPieAttributes
    KDChart::AbstractPieDiagram, 208
    KDChart::PieDiagram, 659
    KDChart::RingDiagram, 804
setPixmap
    KDChart::BackgroundAttributes, 276
setPixmapMode
    KDChart::BackgroundAttributes, 276
setPosition
    KDChart::CartesianAxis, 357
    KDChart::HeaderFooter, 474
    KDChart::Legend, 526
    PrerenderedElement, 750
    PrerenderedLabel, 757
setPositivePosition
    KDChart::DataValueAttributes, 445
setPowerOfTenDivisor
    KDChart::DataValueAttributes, 445
setPrefix
    KDChart::DataValueAttributes, 445
setRainbowColors
    KDChart::Legend, 526
setRectangle
    KDChart::PaintContext, 616
setReferenceArea
    KDChart::Legend, 527
    KDChart::Measure, 612
    KDChart::RelativePosition, 769
setReferenceCoordinatePlane
    KDChart::AbstractCoordinatePlane, 136
    KDChart::CartesianCoordinatePlane, 401
    KDChart::PolarCoordinatePlane, 691
setReferenceDiagram
    KDChart::AbstractCartesianDiagram, 110
    KDChart::BarDiagram, 319
    KDChart::LineDiagram, 579
setReferenceOrientation
    KDChart::Measure, 612
setReferencePoint
    PrerenderedElement, 751
    PrerenderedLabel, 757
setReferencePoints
    KDChart::RelativePosition, 769
setReferencePosition
    KDChart::RelativePosition, 770
setRelativeMode
    KDChart::Measure, 612
setRelativeThickness
    KDChart::RingDiagram, 804
setRootIndex
    KDChart::AbstractCartesianDiagram, 110
    KDChart::AbstractDiagram, 169
    KDChart::AbstractPieDiagram, 209
    KDChart::AbstractPolarDiagram, 243
    KDChart::BarDiagram, 319
    KDChart::LineDiagram, 579
    KDChart::PieDiagram, 659
    KDChart::PolarDiagram, 730
    KDChart::RingDiagram, 805
setRotateCircularLabels
    KDChart::PolarDiagram, 731
setRotation
    KDChart::RelativePosition, 770
    KDChart::TextAttributes, 833
setSelection
    KDChart::AbstractCartesianDiagram, 111
    KDChart::AbstractDiagram, 169
    KDChart::AbstractPieDiagram, 209
    KDChart::AbstractPolarDiagram, 244
    KDChart::BarDiagram, 320
    KDChart::LineDiagram, 580
    KDChart::PieDiagram, 659
    KDChart::PolarDiagram, 731
    KDChart::RingDiagram, 805
setShortLabels
    KDChart::AbstractAxis, 75
    KDChart::CartesianAxis, 357
setShowDelimitersAtPosition
    KDChart::PolarDiagram, 731
setShowInfinite
    KDChart::DataValueAttributes, 446
setShowLabelsAtPosition
    KDChart::PolarDiagram, 731
setShowLines
    KDChart::Legend, 527
setShowRepetitiveDataLabels
    KDChart::DataValueAttributes, 446
setSourceColumnCount
    KDChart::DatasetSelectorWidget, 439
setSourceModel
    KDChart::AttributesModel, 266
    KDChart::DatasetProxyModel, 435
setSourceRootIndex

KDChart::DatasetProxyModel, 435
setSpacing
  KDChart::Legend, 527
setStartPosition
  KDChart::AbstractPieDiagram, 209
  KDChart::PieDiagram, 659
  KDChart::PolarCoordinatePlane, 691
  KDChart::RingDiagram, 805
setSubduedColors
  KDChart::Legend, 528
setSubGridPen
  KDChart::GridAttributes, 459
setSubGridVisible
  KDChart::GridAttributes, 459
setSubType
  KDChart::Widget, 875
setSuffix
  KDChart::DataValueAttributes, 446
setText
  KDChart::HeaderFooter, 475
  KDChart::Legend, 528
  KDChart::TextArea, 826
  KDChart::TextLayoutItem, 841
  PrerenderedLabel, 757
setTextAttributes
  KDChart::AbstractAxis, 76
  KDChart::CartesianAxis, 358
  KDChart::DataValueAttributes, 446
  KDChart::HeaderFooter, 475
  KDChart::Legend, 529
  KDChart::TextArea, 826
  KDChart::TextLayoutItem, 841
setThreeDBarAttributes
  KDChart::BarDiagram, 320
setThreeDLineAttributes
  KDChart::LineDiagram, 580
setThreeDPieAttributes
  KDChart::AbstractPieDiagram, 209, 210
  KDChart::PieDiagram, 660
  KDChart::RingDiagram, 805, 806
setTitleText
  KDChart::CartesianAxis, 358
  KDChart::Legend, 529
setTitleTextAttributes
  KDChart::CartesianAxis, 358
  KDChart::Legend, 529
setTransparency
  KDChart::LineAttributes, 534
setType
  KDChart::BarDiagram, 321
  KDChart::HeaderFooter, 475
  KDChart::LineDiagram, 581
setUseAutomaticMarkerSize
  KDChart::Legend, 529
setUseFixedBarWidth
  KDChart::BarAttributes, 278
setUseFixedDataValueGap
  KDChart::BarAttributes, 278
setUseFixedValueBlockGap
  KDChart::BarAttributes, 278
setUseShadowColors
  KDChart::ThreeDBarAttributes, 847
  KDChart::ThreeDPieAttributes, 857
setValue
  KDChart::Measure, 613
setVerticalHeaderDataMap
  KDChart::AttributesModel, 266
setVerticalPadding
  KDChart::RelativePosition, 770
setVerticalRange
  KDChart::CartesianCoordinatePlane, 401
setVisible
  KDChart::BackgroundAttributes, 276
  KDChart::DataValueAttributes, 446
  KDChart::FrameAttributes, 453
  KDChart::Legend, 530
  KDChart::MarkerAttributes, 601
  KDChart::TextAttributes, 833
setZeroDegreePosition
  KDChart::PolarDiagram, 731
setZeroLinePen
  KDChart::GridAttributes, 459
setZoomCenter
  KDChart::AbstractCoordinatePlane, 136
  KDChart::CartesianCoordinatePlane, 402
  KDChart::PolarCoordinatePlane, 692
setZoomFactorX
  KDChart::AbstractCoordinatePlane, 137
  KDChart::CartesianCoordinatePlane, 402
  KDChart::PolarCoordinatePlane, 692
setZoomFactorY
  KDChart::AbstractCoordinatePlane, 137
  KDChart::CartesianCoordinatePlane, 402
  KDChart::PolarCoordinatePlane, 692
shortLabels
  KDChart::AbstractAxis, 76
  KDChart::CartesianAxis, 359
showDelimitersAtPosition
  KDChart::PolarDiagram, 732
showInfinite
  KDChart::DataValueAttributes, 446
showLabelsAtPosition
  KDChart::PolarDiagram, 732
showLines
  KDChart::Legend, 530
showRepetitiveDataLabels
  KDChart::DataValueAttributes, 447
SignalCompressor

KDChart::SignalCompressor, 811
size
  KDChart::Palette, 619
sizeHint
  KDChart::AbstractCoordinatePlane, 137
  KDChart::AutoSpacerLayoutItem, 272
  KDChart::CartesianAxis, 359
  KDChart::CartesianCoordinatePlane, 403
  KDChart::HeaderFooter, 476
  KDChart::HorizontalLineLayoutItem, 482
  KDChart::Legend, 530
  KDChart::LineLayoutItem, 591
  KDChart::LineWithMarkerLayoutItem, 597
  KDChart::MarkerLayoutItem, 606
  KDChart::PolarCoordinatePlane, 693
  KDChart::TextArea, 827
  KDChart::TextLayoutItem, 842
  KDChart::VerticalLineLayoutItem, 861
  KDTextDocument, 495
sizeHintChanged
  KDChart::AbstractArea, 43
  KDChart::AbstractAxis, 76
  KDChart::AbstractCoordinatePlane, 137
  KDChart::AbstractLayoutItem, 175
  KDChart::AutoSpacerLayoutItem, 273
  KDChart::CartesianAxis, 359
  KDChart::CartesianCoordinatePlane, 403
  KDChart::HeaderFooter, 476
  KDChart::HorizontalLineLayoutItem, 482
  KDChart::LineLayoutItem, 591
  KDChart::LineWithMarkerLayoutItem, 597
  KDChart::MarkerLayoutItem, 606
  KDChart::PolarCoordinatePlane, 693
  KDChart::TextArea, 827
  KDChart::TextLayoutItem, 842
  KDChart::VerticalLineLayoutItem, 861
sizeOfArea
  KDChart::Measure, 613
sizePolicy
  KDChart::AbstractCoordinatePlane, 138
  KDChart::CartesianCoordinatePlane, 403
  KDChart::PolarCoordinatePlane, 693
slotAttributesChanged
  KDChart::DiagramObserver, 450
slotDataChanged
  KDChart::DiagramObserver, 450
slotDataHidden
  KDChart::DiagramObserver, 450
slotHeaderDataChanged
  KDChart::DiagramObserver, 451
slotModelsChanged
  KDChart::DiagramObserver, 451
South
  KDChart::Position, 743
SouthEast
  KDChart::Position, 743
SouthWest
  KDChart::Position, 743
spacing
  KDChart::Legend, 530
Stacked
  KDChart::BarDiagram, 285
  KDChart::LineDiagram, 541
  KDChart::Widget, 866
start
  KDChart::DataDimension, 429
startPosition
  KDChart::AbstractPieDiagram, 210
  KDChart::PieDiagram, 660
  KDChart::PolarCoordinatePlane, 693
  KDChart::RingDiagram, 806
staticPositionCenter
  KDChartPosition.cpp, 974
staticPositionEast
  KDChartPosition.cpp, 974
staticPositionFloating
  KDChartPosition.cpp, 974
staticPositionNames
  KDChartPosition.cpp, 974
staticPositionNorth
  KDChartPosition.cpp, 974
staticPositionNorthEast
  KDChartPosition.cpp, 974
staticPositionNorthWest
  KDChartPosition.cpp, 974
staticPositionSouth
  KDChartPosition.cpp, 975
staticPositionSouthEast
  KDChartPosition.cpp, 975
staticPositionSouthWest
  KDChartPosition.cpp, 975
staticPositionUnknown
  KDChartPosition.cpp, 975
staticPositionWest
  KDChartPosition.cpp, 975
stepWidth
  KDChart::DataDimension, 429
stringToGranularitySequence
  KDChartEnums, 491
stringToLayoutPolicy
  KDChartEnums, 492
stringToMeasureCalculationMode
  KDChartEnums, 492
stringToMeasureOrientation
  KDChartEnums, 493
subduedPalette

KDChart::Palette, 619
subGridPen
 KDChart::GridAttributes, 459
subStepWidth
 KDChart::DataDimension, 429
SubType
 KDChart::Widget, 866
subType
 KDChart::Widget, 876, 877
suffix
 KDChart::DataValueAttributes, 447

takeAxis
 KDChart::AbstractCartesianDiagram, 111
 KDChart::BarDiagram, 321
 KDChart::LineDiagram, 581
takeCoordinatePlane
 KDChart::Chart, 424
takeDiagram
 KDChart::AbstractCoordinatePlane, 138
 KDChart::CartesianCoordinatePlane, 404
 KDChart::PolarCoordinatePlane, 694
takeHeaderFooter
 KDChart::Chart, 425
 KDChart::Widget, 876
takeLegend
 KDChart::Chart, 425
 KDChart::Widget, 876
TEST_SUB_TYPE
 KDChartWidget.cpp, 997
text
 KDChart::HeaderFooter, 476
 KDChart::Legend, 531
 KDChart::TextArea, 827
 KDChart::TextLayoutItem, 842
 PrerenderedLabel, 757
TextArea
 KDChart::TextArea, 815
TextAttributes, 30
 KDChart::TextAttributes, 830
textAttributes
 KDChart::AbstractAxis, 77
 KDChart::CartesianAxis, 360
 KDChart::DataValueAttributes, 447
 KDChart::HeaderFooter, 477
 KDChart::Legend, 531
 KDChart::TextArea, 828
 KDChart::TextLayoutItem, 842
TextLayoutItem
 KDChart::TextLayoutItem, 835
TextLayoutPolicy
 KDChartEnums, 489
texts
 KDChart::Legend, 531
ThreeDAttributesRole
 KDChart, 24
ThreeDBarAttributes
 KDChart::ThreeDBarAttributes, 844
threeDBarAttributes
 KDChart::BarDiagram, 321, 322
ThreeDBarAttributesRole
 KDChart, 24
threeDItemDepth
 KDChart::AbstractCartesianDiagram, 111
 KDChart::BarDiagram, 322
 KDChart::LineDiagram, 582
ThreeDLineAttributes
 KDChart::ThreeDLineAttributes, 849
threeDLineAttributes
 KDChart::LineDiagram, 582, 583
ThreeDLineAttributesRole
 KDChart, 24
ThreeDPieAttributes
 KDChart::ThreeDPieAttributes, 854
threeDPieAttributes
 KDChart::AbstractPieDiagram, 210, 211
 KDChart::PieDiagram, 661
 KDChart::RingDiagram, 806, 807
ThreeDPieAttributesRole
 KDChart, 24
tickLength
 KDChart::CartesianAxis, 360
titleText
 KDChart::CartesianAxis, 360
 KDChart::Legend, 531
titleTextAttributes
 KDChart::CartesianAxis, 361
 KDChart::Legend, 532
Top
 KDChart::CartesianAxis, 329
top
 KDChart::Widget, 878
topOverlap
 KDChart::AbstractArea, 43
 KDChart::AbstractAxis, 77
 KDChart::AbstractCoordinatePlane, 138
 KDChart::CartesianAxis, 361
 KDChart::CartesianCoordinatePlane, 404
 KDChart::PolarCoordinatePlane, 694
translate
 KDChart::AbstractCoordinatePlane, 139
 KDChart::CartesianCoordinatePlane, 404
 KDChart::PolarCoordinatePlane, 695
translateBack
 KDChart::CartesianCoordinatePlane, 405
translatePolar
 KDChart::PolarCoordinatePlane, 695
transparency

KDChart::LineAttributes, 534
type
  KDChart::BarDiagram, 323
  KDChart::HeaderFooter, 477
  KDChart::LineDiagram, 583
  KDChart::Widget, 876

Ui, 31
Unknown
  KDChart::Position, 744
update
  KDChart::AbstractCartesianDiagram, 111
  KDChart::AbstractDiagram, 169
  KDChart::AbstractPieDiagram, 211
  KDChart::AbstractPolarDiagram, 244
  KDChart::BarDiagram, 323
  KDChart::LineDiagram, 583
  KDChart::PieDiagram, 661
  KDChart::PolarDiagram, 732
  KDChart::RingDiagram, 807
updateCommonBrush
  KDChartLayoutItems.cpp, 946
useAutomaticMarkerSize
  KDChart::Legend, 532
useDefaultColors
  KDChart::AbstractCartesianDiagram, 112
  KDChart::AbstractDiagram, 170
  KDChart::AbstractPieDiagram, 211
  KDChart::AbstractPolarDiagram, 244
  KDChart::BarDiagram, 323
  KDChart::LineDiagram, 583
  KDChart::PieDiagram, 662
  KDChart::PolarDiagram, 732
  KDChart::RingDiagram, 807
useFixedBarWidth
  KDChart::BarAttributes, 278
useFixedDataValueGap
  KDChart::BarAttributes, 278
useFixedValueBlockGap
  KDChart::BarAttributes, 278
useRainbowColors
  KDChart::AbstractCartesianDiagram, 112
  KDChart::AbstractDiagram, 170
  KDChart::AbstractPieDiagram, 211
  KDChart::AbstractPolarDiagram, 244
  KDChart::BarDiagram, 323
  KDChart::LineDiagram, 583
  KDChart::PieDiagram, 662
  KDChart::PolarDiagram, 732
  KDChart::RingDiagram, 807
usesExternalAttributesModel
  KDChart::AbstractCartesianDiagram, 112
  KDChart::AbstractDiagram, 170
  KDChart::AbstractPieDiagram, 212
  KDChart::AbstractPolarDiagram, 244
  KDChart::BarDiagram, 324
  KDChart::LineDiagram, 584
  KDChart::PieDiagram, 662
  KDChart::PolarDiagram, 733
  KDChart::RingDiagram, 808
useShadowColors
  KDChart::ThreeDBarAttributes, 847
  KDChart::ThreeDPieAttributes, 857
useSubduedColors
  KDChart::AbstractCartesianDiagram, 112
  KDChart::AbstractDiagram, 170
  KDChart::AbstractPieDiagram, 212
  KDChart::AbstractPolarDiagram, 245
  KDChart::BarDiagram, 324
  KDChart::LineDiagram, 584
  KDChart::PieDiagram, 662
  KDChart::PolarDiagram, 733
  KDChart::RingDiagram, 808

validDepth
  KDChart::AbstractThreeDAttributes, 251
  KDChart::ThreeDBarAttributes, 847
  KDChart::ThreeDLineAttributes, 852
  KDChart::ThreeDPieAttributes, 857
value
  KDChart::Measure, 613
  KDChart::Position, 742
valueForCell
  KDChart::AbstractCartesianDiagram, 113
  KDChart::AbstractDiagram, 171
  KDChart::AbstractPieDiagram, 212
  KDChart::AbstractPolarDiagram, 245
  KDChart::BarDiagram, 324
  KDChart::LineDiagram, 584
  KDChart::PieDiagram, 663
  KDChart::PolarDiagram, 733
  KDChart::RingDiagram, 808
valueForCellTesting
  KDChart::LineDiagram, 585
valueTotals
  KDChart::AbstractPieDiagram, 212
  KDChart::AbstractPolarDiagram, 245
  KDChart::PieDiagram, 663
  KDChart::PolarDiagram, 734
  KDChart::RingDiagram, 809
verticalHeaderDataMap
  KDChart::AttributesModel, 266
VerticalLineLayoutItem
  KDChart::VerticalLineLayoutItem, 858
verticalOffset
  KDChart::AbstractCartesianDiagram, 113
  KDChart::AbstractDiagram, 171
  KDChart::AbstractPieDiagram, 213

  KDChart::AbstractPolarDiagram, 246
  KDChart::BarDiagram, 324
  KDChart::LineDiagram, 585
  KDChart::PieDiagram, 663
  KDChart::PolarDiagram, 734
  KDChart::RingDiagram, 809
verticalPadding
  KDChart::RelativePosition, 770
verticalRange
  KDChart::CartesianCoordinatePlane, 405
visualRect
  KDChart::AbstractCartesianDiagram, 113
  KDChart::AbstractDiagram, 171
  KDChart::AbstractPieDiagram, 213
  KDChart::AbstractPolarDiagram, 246
  KDChart::BarDiagram, 325
  KDChart::LineDiagram, 585
  KDChart::PieDiagram, 664
  KDChart::PolarDiagram, 734
  KDChart::RingDiagram, 809
visualRegionForSelection
  KDChart::AbstractCartesianDiagram, 113
  KDChart::AbstractDiagram, 171
  KDChart::AbstractPieDiagram, 213
  KDChart::AbstractPolarDiagram, 246
  KDChart::BarDiagram, 325
  KDChart::LineDiagram, 585
  KDChart::PieDiagram, 664
  KDChart::PolarDiagram, 734
  KDChart::RingDiagram, 809

West
  KDChart::Position, 744
Widget
  KDChart::Widget, 866

xCenter
  KDChart::ZoomParameters, 880
xFactor
  KDChart::ZoomParameters, 880

yCenter
  KDChart::ZoomParameters, 880
yFactor
  KDChart::ZoomParameters, 880

zeroDegreePosition
  KDChart::PolarDiagram, 734
zeroLinePen
  KDChart::GridAttributes, 459
zoomCenter
  KDChart::AbstractCoordinatePlane, 139
  KDChart::CartesianCoordinatePlane, 405
  KDChart::PolarCoordinatePlane, 695
zoomFactorX
  KDChart::AbstractCoordinatePlane, 139
  KDChart::CartesianCoordinatePlane, 406
  KDChart::PolarCoordinatePlane, 696
zoomFactorY
  KDChart::AbstractCoordinatePlane, 140
  KDChart::CartesianCoordinatePlane, 406
  KDChart::PolarCoordinatePlane, 696
ZoomParameters
  KDChart::ZoomParameters, 879