Qwt User's Guide 6.1.1

Generated by Doxygen 1.8.5

Thu Sep 18 2014 16:59:51

ii CONTENTS

Contents

| | 1.1 1.2 | License | 2 |
|---|------------|--|----|
| | 1.2 | | _ |
| | | Platforms | 2 |
| | 1.3 | What's new | 2 |
| | 1.4 | Screenshots | 2 |
| | 1.5 | Downloads | 2 |
| | 1.6 | Installation | 2 |
| | 1.7 | Support | 3 |
| | 1.8 | Related Projects | 3 |
| | 1.9 | Donations | 3 |
| | 1.10 | Credits: | 3 |
| 2 | Wha | t's new in Qwt 6.1 | 3 |
| | 2.1 | New plot items | 3 |
| | 2.2 | Scales beyond linear and logarithmic transformations | 4 |
| | | 2.2.1 Datetime scales | 4 |
| | 2.3 | Redesign of the dial and meter widgets | 4 |
| | 2.4 | Basic support for an OpenGL plot canvas | 5 |
| | 2.5 | A new system for plot legends | 5 |
| | 2.6 | Off-screen paint device for vector graphics | 5 |
| | 2.7 | QwtWidgetOverlay | 5 |
| | 2.8 | QwtSymbol | 5 |
| | 2.9 | QwtPlotCurve | 6 |
| | 2.10 | QwtPlot | 6 |
| | 2.11 | Other | 6 |
| | | 2.11.1 QwtScaleDiv | 6 |
| | | 2.11.2 QwtScaleEngine | 7 |
| | | 2.11.3 QwtPlotLayout | 7 |
| | | 2.11.4 QwtPlotCanvas | 7 |
| | | 2.11.5 Other changes | 7 |
| | 2.12 | Summary of the new classes | 8 |
| 3 | Insta | illing Qwt | 8 |
| | 3.1 | Download | 8 |
| | 3.2 | Installing Qwt | 9 |
| | | 3.2.1 Configuration | 9 |
| | | 3.2.2 Build and installation | 10 |
| | 3.3 | Qwt and the Qt tool chain | 11 |
| | | 3.3.1 Designer plugin | 11 |

| | | 3.3.2 | Online Help | 11 |
|----|-------|------------------|--|----------|
| | 3.4 | Buildin | g a Qwt application | 12 |
| | 3.5 | Runnin | ng a Qwt application | 12 |
| | | 3.5.1 | Windows | 12 |
| | | 3.5.2 | GNU/Linux | 12 |
| 4 | Qwt | License | e, Version 1.0 | 13 |
| 5 | Curv | e Plots | | 19 |
| 6 | Scat | ter Plot | | 19 |
| 7 | Spec | trogran | m, Contour Plot | 19 |
| 8 | Histo | ogram | | 19 |
| 9 | Dials | s, Comp | passes, Knobs, Wheels, Sliders, Thermos | 19 |
| 10 | Hiera | archical | l Index | 19 |
| | | | Hierarchy | 19 |
| 11 | Clas | s Index | | 24 |
| | 11.1 | Class I | _ist | 24 |
| 12 | Clas | s Docu | mentation | 31 |
| | 12.1 | QwtEv | entPattern::KeyPattern Class Reference | 31 |
| | | 12.1.1 | Detailed Description | 31 |
| | 12.2 | QwtEv | entPattern::MousePattern Class Reference | 31 |
| | | 12.2.1 | Detailed Description | 32 |
| | 12.3 | QwtAb | stractLegend Class Reference | 32 |
| | | 12.3.1 | Detailed Description | 33 |
| | | | Constructor & Destructor Documentation | 33 |
| | | | Member Function Documentation | 33 |
| | 12.4 | | stractScale Class Reference | 34 |
| | | | Detailed Description | 35 |
| | | | Constructor & Destructor Documentation | 36 |
| | 10.5 | | Member Function Documentation | 37 |
| | 12.5 | | stractScaleDraw Class Reference | 42 43 |
| | | 12.5.1 12.5.2 | · | 43 |
| | | 12.5.2 | | 43 |
| | | | Member Function Documentation | 44 |
| | 12.6 | | stractSeriesStore Class Reference | 50 |
| | _,• | | Detailed Description | 51 |
| | | | • | ·- |

iv CONTENTS

| 12.6.2 Member Function Documentation | 51 |
|---|----|
| 12.7 QwtAbstractSlider Class Reference | 52 |
| 12.7.1 Detailed Description | 54 |
| 12.7.2 Constructor & Destructor Documentation | 54 |
| 12.7.3 Member Function Documentation | 54 |
| 12.8 QwtAlphaColorMap Class Reference | 61 |
| 12.8.1 Detailed Description | 62 |
| 12.8.2 Constructor & Destructor Documentation | 62 |
| 12.8.3 Member Function Documentation | 62 |
| 12.9 QwtAnalogClock Class Reference | 63 |
| 12.9.1 Detailed Description | 64 |
| 12.9.2 Member Enumeration Documentation | 64 |
| 12.9.3 Constructor & Destructor Documentation | 65 |
| 12.9.4 Member Function Documentation | 66 |
| 12.10QwtArraySeriesData< T > Class Template Reference | 67 |
| 12.10.1 Detailed Description | 68 |
| 12.10.2 Constructor & Destructor Documentation | 68 |
| 12.10.3 Member Function Documentation | 68 |
| 12.11 QwtArrowButton Class Reference | 69 |
| 12.11.1 Detailed Description | 70 |
| 12.11.2 Constructor & Destructor Documentation | 70 |
| 12.11.3 Member Function Documentation | 70 |
| 12.12QwtClipper Class Reference | 71 |
| 12.12.1 Detailed Description | 71 |
| 12.12.2 Member Function Documentation | 71 |
| 12.13QwtColorMap Class Reference | 72 |
| 12.13.1 Detailed Description | 73 |
| 12.13.2 Member Enumeration Documentation | 73 |
| 12.13.3 Member Function Documentation | 74 |
| 12.14QwtColumnRect Class Reference | 75 |
| 12.14.1 Detailed Description | 75 |
| 12.14.2 Member Enumeration Documentation | 76 |
| 12.14.3 Member Function Documentation | 76 |
| 12.15QwtColumnSymbol Class Reference | 76 |
| 12.15.1 Detailed Description | 77 |
| 12.15.2 Member Enumeration Documentation | 77 |
| 12.15.3 Constructor & Destructor Documentation | 77 |
| 12.15.4 Member Function Documentation | 77 |
| 12.16QwtCompass Class Reference | 79 |
| 12.16.1 Detailed Description | 80 |

| 12.16.2 Constructor & Destructor Documentation | 81 |
|--|-----|
| 12.16.3 Member Function Documentation | 81 |
| 12.17QwtCompassMagnetNeedle Class Reference | 82 |
| 12.17.1 Detailed Description | 83 |
| 12.17.2 Member Enumeration Documentation | 83 |
| 12.17.3 Member Function Documentation | 83 |
| 12.18QwtCompassRose Class Reference | 84 |
| 12.18.1 Detailed Description | 84 |
| 12.18.2 Member Function Documentation | 84 |
| 12.19QwtCompassScaleDraw Class Reference | 85 |
| 12.19.1 Detailed Description | 85 |
| 12.19.2 Constructor & Destructor Documentation | 86 |
| 12.19.3 Member Function Documentation | 86 |
| 12.20QwtCompassWindArrow Class Reference | 87 |
| 12.20.1 Detailed Description | 87 |
| 12.20.2 Member Enumeration Documentation | 88 |
| 12.20.3 Constructor & Destructor Documentation | 88 |
| 12.20.4 Member Function Documentation | 88 |
| 12.21 QwtCounter Class Reference | 88 |
| 12.21.1 Detailed Description | 90 |
| 12.21.2 Member Enumeration Documentation | 90 |
| 12.21.3 Constructor & Destructor Documentation | 91 |
| 12.21.4 Member Function Documentation | 91 |
| 12.22QwtCPointerData Class Reference | 97 |
| 12.22.1 Detailed Description | 97 |
| 12.22.2 Constructor & Destructor Documentation | 97 |
| 12.22.3 Member Function Documentation | 98 |
| 12.23QwtCurveFitter Class Reference | 99 |
| 12.23.1 Detailed Description | 99 |
| 12.23.2 Member Function Documentation | 99 |
| 12.24QwtDate Class Reference | 100 |
| 12.24.1 Detailed Description | 100 |
| 12.24.2 Member Enumeration Documentation | 100 |
| 12.24.3 Member Function Documentation | 101 |
| 12.25QwtDateScaleDraw Class Reference | 105 |
| 12.25.1 Detailed Description | 106 |
| 12.25.2 Constructor & Destructor Documentation | 106 |
| 12.25.3 Member Function Documentation | 107 |
| 12.26QwtDateScaleEngine Class Reference | 110 |
| 12.26.1 Detailed Description | 111 |

vi CONTENTS

| 12.26.2 Constructor & Destructor Documentation |
|--|
| 12.26.3 Member Function Documentation |
| 12.27QwtDial Class Reference |
| 12.27.1 Detailed Description |
| 12.27.2 Member Enumeration Documentation |
| 12.27.3 Constructor & Destructor Documentation |
| 12.27.4 Member Function Documentation |
| 12.28QwtDialNeedle Class Reference |
| 12.28.1 Detailed Description |
| 12.28.2 Member Function Documentation |
| 12.29QwtDialSimpleNeedle Class Reference |
| 12.29.1 Detailed Description |
| 12.29.2 Member Enumeration Documentation |
| 12.29.3 Constructor & Destructor Documentation |
| 12.29.4 Member Function Documentation |
| 12.30QwtDynGridLayout Class Reference |
| 12.30.1 Detailed Description |
| 12.30.2 Constructor & Destructor Documentation |
| 12.30.3 Member Function Documentation |
| 12.31 QwtEventPattern Class Reference |
| 12.31.1 Detailed Description |
| 12.31.2 Member Enumeration Documentation |
| 12.31.3 Constructor & Destructor Documentation |
| 12.31.4 Member Function Documentation |
| 12.32QwtGraphic Class Reference |
| 12.32.1 Detailed Description |
| 12.32.2 Member Typedef Documentation |
| 12.32.3 Member Enumeration Documentation |
| 12.32.4 Constructor & Destructor Documentation |
| 12.32.5 Member Function Documentation |
| 12.33 QwtInterval Class Reference |
| 12.33.1 Detailed Description |
| 12.33.2 Member Enumeration Documentation |
| 12.33.3 Constructor & Destructor Documentation |
| 12.33.4 Member Function Documentation |
| 12.34QwtIntervalSample Class Reference |
| 12.34.1 Detailed Description |
| 12.34.2 Constructor & Destructor Documentation |
| 12.35QwtIntervalSeriesData Class Reference |
| 12.35.1 Detailed Description |

CONTENTS vii

| 12.35.2 Constructor & Destructor Documentation |
|--|
| 12.35.3 Member Function Documentation |
| 12.36QwtIntervalSymbol Class Reference |
| 12.36.1 Detailed Description |
| 12.36.2 Member Enumeration Documentation |
| 12.36.3 Constructor & Destructor Documentation |
| 12.36.4 Member Function Documentation |
| 12.37QwtKnob Class Reference |
| 12.37.1 Detailed Description |
| 12.37.2 Member Enumeration Documentation |
| 12.37.3 Constructor & Destructor Documentation |
| 12.37.4 Member Function Documentation |
| 12.38QwtLegend Class Reference |
| 12.38.1 Detailed Description |
| 12.38.2 Constructor & Destructor Documentation |
| 12.38.3 Member Function Documentation |
| 12.39QwtLegendData Class Reference |
| 12.39.1 Detailed Description |
| 12.39.2 Member Enumeration Documentation |
| 12.39.3 Member Function Documentation |
| 12.40 QwtLegendLabel Class Reference |
| 12.40.1 Detailed Description |
| 12.40.2 Constructor & Destructor Documentation |
| 12.40.3 Member Function Documentation |
| 12.41 QwtLinearColorMap Class Reference |
| 12.41.1 Detailed Description |
| 12.41.2 Member Enumeration Documentation |
| 12.41.3 Constructor & Destructor Documentation |
| 12.41.4 Member Function Documentation |
| 12.42QwtLinearScaleEngine Class Reference |
| 12.42.1 Detailed Description |
| 12.42.2 Constructor & Destructor Documentation |
| 12.42.3 Member Function Documentation |
| 12.43 QwtLogScaleEngine Class Reference |
| 12.43.1 Detailed Description |
| 12.43.2 Constructor & Destructor Documentation |
| 12.43.3 Member Function Documentation |
| 12.44QwtLogTransform Class Reference |
| 12.44.1 Detailed Description |
| 12.44.2 Member Function Documentation |

viii CONTENTS

| 12.45 QwtMagnifier Class Reference |
|--|
| 12.45.1 Detailed Description |
| 12.45.2 Constructor & Destructor Documentation |
| 12.45.3 Member Function Documentation |
| 12.46QwtMathMLTextEngine Class Reference |
| 12.46.1 Detailed Description |
| 12.46.2 Member Function Documentation |
| 12.47QwtMatrixRasterData Class Reference |
| 12.47.1 Detailed Description |
| 12.47.2 Member Enumeration Documentation |
| 12.47.3 Member Function Documentation |
| 12.48QwtNullPaintDevice Class Reference |
| 12.48.1 Detailed Description |
| 12.48.2 Member Enumeration Documentation |
| 12.48.3 Member Function Documentation |
| 12.49QwtNullTransform Class Reference |
| 12.49.1 Detailed Description |
| 12.49.2 Member Function Documentation |
| 12.50QwtOHLCSample Class Reference |
| 12.50.1 Detailed Description |
| 12.50.2 Constructor & Destructor Documentation |
| 12.50.3 Member Function Documentation |
| 12.50.4 Member Data Documentation |
| 12.51 QwtPainter Class Reference |
| 12.51.1 Detailed Description |
| 12.51.2 Member Function Documentation |
| 12.52QwtPainterCommand Class Reference |
| 12.52.1 Detailed Description |
| 12.52.2 Member Enumeration Documentation |
| 12.52.3 Constructor & Destructor Documentation |
| 12.52.4 Member Function Documentation |
| 12.53 QwtPanner Class Reference |
| 12.53.1 Detailed Description |
| 12.53.2 Constructor & Destructor Documentation |
| 12.53.3 Member Function Documentation |
| 12.54QwtPicker Class Reference |
| 12.54.1 Detailed Description |
| 12.54.2 Member Enumeration Documentation |
| 12.54.3 Constructor & Destructor Documentation |
| 12.54.4 Member Function Documentation |

| 12.55QwtPickerClickPointMachine Class Reference |
|---|
| 12.55.1 Detailed Description |
| 12.56QwtPickerClickRectMachine Class Reference |
| 12.56.1 Detailed Description |
| 12.57QwtPickerDragLineMachine Class Reference |
| 12.57.1 Detailed Description |
| 12.58QwtPickerDragPointMachine Class Reference |
| 12.58.1 Detailed Description |
| 12.59QwtPickerDragRectMachine Class Reference |
| 12.59.1 Detailed Description |
| 12.60 QwtPickerMachine Class Reference |
| 12.60.1 Detailed Description |
| 12.60.2 Member Enumeration Documentation |
| 12.61 QwtPickerPolygonMachine Class Reference |
| 12.61.1 Detailed Description |
| 12.62QwtPickerTrackerMachine Class Reference |
| 12.62.1 Detailed Description |
| 12.63QwtPixelMatrix Class Reference |
| 12.63.1 Detailed Description |
| 12.63.2 Constructor & Destructor Documentation |
| 12.63.3 Member Function Documentation |
| 12.64QwtPlainTextEngine Class Reference |
| 12.64.1 Detailed Description |
| 12.64.2 Member Function Documentation |
| 12.65QwtPlot Class Reference |
| 12.65.1 Detailed Description |
| 12.65.2 Member Enumeration Documentation |
| 12.65.3 Constructor & Destructor Documentation |
| 12.65.4 Member Function Documentation |
| 12.66QwtPlotAbstractBarChart Class Reference |
| 12.66.1 Detailed Description |
| 12.66.2 Member Enumeration Documentation |
| 12.66.3 Constructor & Destructor Documentation |
| 12.66.4 Member Function Documentation |
| 12.67QwtPlotBarChart Class Reference |
| 12.67.1 Detailed Description |
| 12.67.2 Member Enumeration Documentation |
| 12.67.3 Constructor & Destructor Documentation |
| 12.67.4 Member Function Documentation |
| 12.68QwtPlotCanvas Class Reference |

X CONTENTS

| 12.68.1 Detailed Description | 297 |
|--|-----|
| 12.68.2 Member Enumeration Documentation | 297 |
| 12.68.3 Constructor & Destructor Documentation | 298 |
| 12.68.4 Member Function Documentation | 299 |
| 12.69QwtPlotCurve Class Reference | 302 |
| 12.69.1 Detailed Description | 304 |
| 12.69.2 Member Enumeration Documentation | 304 |
| 12.69.3 Constructor & Destructor Documentation | 306 |
| 12.69.4 Member Function Documentation | 307 |
| 12.70QwtPlotDict Class Reference | 316 |
| 12.70.1 Detailed Description | 316 |
| 12.70.2 Constructor & Destructor Documentation | 316 |
| 12.70.3 Member Function Documentation | 317 |
| 12.71 QwtPlotDirectPainter Class Reference | 318 |
| 12.71.1 Detailed Description | 319 |
| 12.71.2 Member Enumeration Documentation | 319 |
| 12.71.3 Member Function Documentation | 320 |
| 12.72QwtPlotGLCanvas Class Reference | 321 |
| 12.72.1 Detailed Description | 323 |
| 12.72.2 Member Enumeration Documentation | 323 |
| 12.72.3 Constructor & Destructor Documentation | 323 |
| 12.72.4 Member Function Documentation | 324 |
| 12.73QwtPlotGrid Class Reference | 327 |
| 12.73.1 Detailed Description | 328 |
| 12.73.2 Member Function Documentation | 328 |
| 12.74QwtPlotHistogram Class Reference | 332 |
| 12.74.1 Detailed Description | 334 |
| 12.74.2 Member Enumeration Documentation | 334 |
| 12.74.3 Constructor & Destructor Documentation | 334 |
| 12.74.4 Member Function Documentation | 335 |
| 12.75QwtPlotIntervalCurve Class Reference | 341 |
| 12.75.1 Detailed Description | 342 |
| 12.75.2 Member Enumeration Documentation | 342 |
| 12.75.3 Constructor & Destructor Documentation | 343 |
| 12.75.4 Member Function Documentation | 343 |
| 12.76QwtPlotItem Class Reference | 348 |
| 12.76.1 Detailed Description | 351 |
| 12.76.2 Member Enumeration Documentation | 351 |
| 12.76.3 Constructor & Destructor Documentation | |
| 12.76.4 Member Function Documentation | 353 |

CONTENTS xi

| 12.77QwtPlotLayout Class Reference |
|--|
| 12.77.1 Detailed Description |
| 12.77.2 Member Enumeration Documentation |
| 12.77.3 Member Function Documentation |
| 12.78QwtPlotLegendItem Class Reference |
| 12.78.1 Detailed Description |
| 12.78.2 Member Enumeration Documentation |
| 12.78.3 Member Function Documentation |
| 12.79QwtPlotMagnifier Class Reference |
| 12.79.1 Detailed Description |
| 12.79.2 Constructor & Destructor Documentation |
| 12.79.3 Member Function Documentation |
| 12.80QwtPlotMarker Class Reference |
| 12.80.1 Detailed Description |
| 12.80.2 Member Enumeration Documentation |
| 12.80.3 Member Function Documentation |
| 12.81QwtPlotMultiBarChart Class Reference |
| 12.81.1 Detailed Description |
| 12.81.2 Member Enumeration Documentation |
| 12.81.3 Constructor & Destructor Documentation |
| 12.81.4 Member Function Documentation |
| 12.82QwtPlotPanner Class Reference |
| 12.82.1 Detailed Description |
| 12.82.2 Constructor & Destructor Documentation |
| 12.82.3 Member Function Documentation |
| 12.83QwtPlotPicker Class Reference |
| 12.83.1 Detailed Description |
| 12.83.2 Constructor & Destructor Documentation |
| 12.83.3 Member Function Documentation |
| 12.84QwtPlotRasterItem Class Reference |
| 12.84.1 Detailed Description |
| 12.84.2 Member Enumeration Documentation |
| 12.84.3 Member Function Documentation |
| 12.85QwtPlotRenderer Class Reference |
| 12.85.1 Detailed Description |
| 12.85.2 Member Enumeration Documentation |
| 12.85.3 Constructor & Destructor Documentation |
| 12.85.4 Member Function Documentation |
| 12.86QwtPlotRescaler Class Reference |
| 12.86.1 Detailed Description |

xii CONTENTS

| 12.86.2 Member Enumeration Documentation |
|--|
| 12.86.3 Constructor & Destructor Documentation |
| 12.86.4 Member Function Documentation |
| 12.87QwtPlotScaleItem Class Reference |
| 12.87.1 Detailed Description |
| 12.87.2 Constructor & Destructor Documentation |
| 12.87.3 Member Function Documentation |
| 12.88QwtPlotSeriesItem Class Reference |
| 12.88.1 Detailed Description |
| 12.88.2 Constructor & Destructor Documentation |
| 12.88.3 Member Function Documentation |
| 12.89QwtPlotShapeItem Class Reference |
| 12.89.1 Detailed Description |
| 12.89.2 Member Enumeration Documentation |
| 12.89.3 Constructor & Destructor Documentation |
| 12.89.4 Member Function Documentation |
| 12.90QwtPlotSpectroCurve Class Reference |
| 12.90.1 Detailed Description |
| 12.90.2 Member Enumeration Documentation |
| 12.90.3 Constructor & Destructor Documentation |
| 12.90.4 Member Function Documentation |
| 12.91 QwtPlotSpectrogram Class Reference |
| 12.91.1 Detailed Description |
| 12.91.2 Member Enumeration Documentation |
| 12.91.3 Constructor & Destructor Documentation |
| 12.91.4 Member Function Documentation |
| 12.92QwtPlotSvgItem Class Reference |
| 12.92.1 Detailed Description |
| 12.92.2 Constructor & Destructor Documentation |
| 12.92.3 Member Function Documentation |
| 12.93QwtPlotTextLabel Class Reference |
| 12.93.1 Detailed Description |
| 12.93.2 Constructor & Destructor Documentation |
| 12.93.3 Member Function Documentation |
| 12.94QwtPlotTradingCurve Class Reference |
| 12.94.1 Detailed Description |
| 12.94.2 Member Enumeration Documentation |
| 12.94.3 Constructor & Destructor Documentation |
| 12.94.4 Member Function Documentation |
| 12.95QwtPlotZoneItem Class Reference |

CONTENTS xiii

| 12.95.1 Detailed Description |
|--|
| 12.95.2 Constructor & Destructor Documentation |
| 12.95.3 Member Function Documentation |
| 12.96QwtPlotZoomer Class Reference |
| 12.96.1 Detailed Description |
| 12.96.2 Constructor & Destructor Documentation |
| 12.96.3 Member Function Documentation |
| 12.97QwtPoint3D Class Reference |
| 12.97.1 Detailed Description |
| 12.97.2 Constructor & Destructor Documentation |
| 12.97.3 Member Function Documentation |
| 12.98QwtPoint3DSeriesData Class Reference |
| 12.98.1 Detailed Description |
| 12.98.2 Constructor & Destructor Documentation |
| 12.98.3 Member Function Documentation |
| 12.99QwtPointArrayData Class Reference |
| 12.99.1 Detailed Description |
| 12.99.2 Constructor & Destructor Documentation |
| 12.99.3 Member Function Documentation |
| 12.10 QwtPointMapper Class Reference |
| 12.100. Detailed Description |
| 12.100.2Member Typedef Documentation |
| 12.100.3Member Enumeration Documentation |
| 12.100.4Member Function Documentation |
| 12.10 QwtPointPolar Class Reference |
| 12.101. Detailed Description |
| 12.101. Constructor & Destructor Documentation |
| 12.101.3Member Function Documentation |
| 12.10 20 wt Point Series Data Class Reference |
| 12.102. Detailed Description |
| 12.102. Constructor & Destructor Documentation |
| 12.102.3Member Function Documentation |
| 12.10®wtPowerTransform Class Reference |
| 12.103. Detailed Description |
| 12.103. Constructor & Destructor Documentation |
| 12.103.3Member Function Documentation |
| 12.10 QwtRasterData Class Reference |
| 12.104. Detailed Description |
| 12.104.2Member Enumeration Documentation |
| 12.104.3 Member Function Documentation |

xiv CONTENTS

| 12.10 QwtRichTextEngine Class Reference |
|---|
| 12.105. Detailed Description |
| 12.105.2Member Function Documentation |
| 12.10 © wtRoundScaleDraw Class Reference |
| 12.106. Detailed Description |
| 12.106. Constructor & Destructor Documentation |
| 12.106.3Member Function Documentation |
| 12.10 QwtSamplingThread Class Reference |
| 12.107. Detailed Description |
| 12.107.2Member Function Documentation |
| 12.10 the With Scale Arithmetic Class Reference |
| 12.108. Detailed Description |
| 12.108.2Member Function Documentation |
| 12.10 QwtScaleDiv Class Reference |
| 12.109. Detailed Description |
| 12.109.2Member Enumeration Documentation |
| 12.109. Constructor & Destructor Documentation |
| 12.109.4Member Function Documentation |
| 12.11 QwtScaleDraw Class Reference |
| 12.110. Detailed Description |
| 12.110.2Member Enumeration Documentation |
| 12.110. Constructor & Destructor Documentation |
| 12.110.4Member Function Documentation |
| 12.11 QwtScaleEngine Class Reference |
| 12.111. Detailed Description |
| 12.111.2Member Enumeration Documentation |
| 12.111. Constructor & Destructor Documentation |
| 12.111.4Member Function Documentation |
| 12.11 QwtScaleMap Class Reference |
| 12.112. Detailed Description |
| 12.112. Constructor & Destructor Documentation |
| 12.112.3Member Function Documentation |
| 12.11 SwtScale Widget Class Reference 540 |
| 12.113. Detailed Description |
| 12.113.2Member Enumeration Documentation |
| 12.113. Constructor & Destructor Documentation |
| 12.113.4Member Function Documentation |
| 12.11 QwtSeries Data < T > Class Template Reference |
| 12.114. Detailed Description |
| 12.114.2 Member Function Documentation |

CONTENTS xv

| 12.11 QwtSeriesStore < T > Class Template Reference |
|---|
| 12.115. Detailed Description |
| 12.115.2Member Function Documentation |
| 12.11 QwtSetSample Class Reference |
| 12.116. Detailed Description |
| 12.116. Constructor & Destructor Documentation |
| 12.116.3 Member Function Documentation |
| 12.11 QwtSetSeriesData Class Reference |
| 12.117. Detailed Description |
| 12.117. Constructor & Destructor Documentation |
| 12.117.3Member Function Documentation |
| 12.11 Newt Simple Compass Rose Class Reference |
| 12.118. Detailed Description |
| 12.118.2Constructor & Destructor Documentation |
| 12.118.3 Member Function Documentation |
| 12.11 QwtSlider Class Reference |
| 12.119. Detailed Description |
| 12.119.2 Member Enumeration Documentation |
| 12.119. Constructor & Destructor Documentation |
| 12.119.4Member Function Documentation |
| 12.12 QwtSpline Class Reference |
| 12.120. Detailed Description |
| 12.120.2Member Enumeration Documentation |
| 12.120. Constructor & Destructor Documentation |
| 12.120.4Member Function Documentation |
| 12.12 QwtSplineCurveFitter Class Reference |
| 12.121. Detailed Description |
| 12.121.2 Member Enumeration Documentation |
| 12.121.3 Member Function Documentation |
| 12.12 DwtSymbol Class Reference |
| 12.122. Detailed Description |
| 12.122.2Member Enumeration Documentation |
| 12.122. Constructor & Destructor Documentation |
| 12.122.4Member Function Documentation |
| 12.12 OwtSyntheticPointData Class Reference |
| 12.123. Detailed Description |
| 12.123. Constructor & Destructor Documentation |
| 12.123.3Member Function Documentation |
| 12.12 QwtSystemClock Class Reference |
| 12.124. Detailed Description |

| 12.124.2Member Function Documentation |) 2 |
|--|----------------|
| 12.12 QwtText Class Reference |) 2 |
| 12.125. Detailed Description |) 4 |
| 12.125.2Member Enumeration Documentation |) 4 |
| 12.125.3Constructor & Destructor Documentation |) 5 |
| 12.125.4Member Function Documentation |) 5 |
| 12.12 QwtTextEngine Class Reference |)1 |
| 12.126. Detailed Description |)1 |
| 12.126.2Member Function Documentation |)2 |
| 12.12 QwtTextLabel Class Reference |)3 |
| 12.127. Detailed Description |)4 |
| 12.127. Constructor & Destructor Documentation |)4 |
| 12.127.3Member Function Documentation |)5 |
| 12.12 QwtThermo Class Reference |)7 |
| 12.128. Detailed Description |)9 |
| 12.128.2 Member Enumeration Documentation |)9 |
| 12.128. Constructor & Destructor Documentation | 10 |
| 12.128.4Member Function Documentation | 10 |
| 12.12 QwtTradingChartData Class Reference | 19 |
| 12.129. Detailed Description | 19 |
| 12.129. Constructor & Destructor Documentation | 20 |
| 12.129.3Member Function Documentation | 20 |
| 12.13 QwtTransform Class Reference | 20 |
| 12.130. Detailed Description | 21 |
| 12.130.2Member Function Documentation | 21 |
| 12.13 QwtWeeding Curve Fitter Class Reference | 22 |
| 12.131. Detailed Description | 22 |
| 12.131. Constructor & Destructor Documentation | 23 |
| 12.131.3 Member Function Documentation | 23 |
| 12.13 DwtWheel Class Reference | 24 |
| 12.132. Detailed Description | 26 |
| 12.132.2 Member Function Documentation | 27 |
| 12.13 QwtWidgetOverlay Class Reference | 37 |
| 12.133. Detailed Description | 39 |
| 12.133.2Member Enumeration Documentation | 39 |
| 12.133. Constructor & Destructor Documentation | 40 |
| 12.133.4Member Function Documentation | 40 |
| | |

1 Qwt - Qt Widgets for Technical Applications

The Qwt library contains GUI Components and utility classes which are primarily useful for programs with a technical background. Beside a framework for 2D plots it provides scales, sliders, dials, compasses, thermometers, wheels and knobs to control or display values, arrays, or ranges of type double.

1.1 License

Qwt is distributed under the terms of the Qwt License, Version 1.0.

1.2 Platforms

Qwt 6.1 might be usable in all environments where you find Qt. It is compatible with Qt4 (>= 4.4) and Qt5.

1.3 What's new

Read the summary of the most important changes.

1.4 Screenshots

- Curve Plots
- Scatter Plot
- Spectrogram, Contour Plot
- · Histogram
- Dials, Compasses, Knobs, Wheels, Sliders, Thermos Screenshots are only available in the HTML docs.

1.5 Downloads

Stable releases or prereleases are available at the Qwt project page.

For getting a snapshot with all bugfixes for the latest 5.2 release:

```
svn checkout svn://svn.code.sf.net/p/qwt/code/branches/qwt-5.2
```

For getting a snapshot with all bugfixes for the latest 6.1 release:

```
svn checkout svn://svn.code.sf.net/p/qwt/code/branches/qwt-6.1
```

For getting a development snapshot from the SVN repository:

```
svn checkout svn://svn.code.sf.net/p/qwt/code/trunk/qwt
```

1.6 Installation

Qwt doesn't distribute binary packages, but today all major Linux distributors offer one. Note, that these packages often don't include the examples.

When no binary packages are available (f.e. on Windows) Qwt needs to be compiled and installed on the target system.

1.7 Support

1.7 Support

· Mailing list

For all kind of Qwt related questions use the Qwt mailing list.

If you prefer newsgroups use the mail to news gateway of Gmane.

Forum

Qt Centre is a great resource for Qt related questions. It has a sub forum, that is dedicated to Qwt related questions.

· Individual support

If you are looking for individual support, or need someone who implements your Qwt component/application contact support@qwt-project.org. Sending requests to this address without a good reason for not using public support channels might be silently ignored.

1.8 Related Projects

```
QwtPolar, a polar plot widget.
```

QwtPlot3D, an OpenGL 3D plot widget.

1.9 Donations

Sourceforge offers a Donation System via PayPal. You can use it, if you like to support the development of Qwt.

1.10 Credits:

Authors:

Uwe Rathmann, Josef Wilgen (<= Qwt 0.2)

Project admin:

Uwe Rathmann < rathmann@users.sourceforge.net>

2 What's new in Qwt 6.1

2.1 New plot items

QwtPlotBarChart

Bar chart, see "examples/distrowatch"

· QwtPlotMultiBarChart

Chart of grouped bars - stacked or aligned side by side. See "examples/barchart"

QwtPlotTradingCurve

Candlestick or OHLC charts typically used to describe price movements over time. See "examples/stockchart"

QwtPlotShapeItem

A plot item to display rectangles, circles, polygons and all other type of shapes (built from intersections or unifications), that can be expressed by a QPainterPath. See "examples/itemeditor"

QwtPlotLegendItem

A legend on the plot canvas. See "examples/legends"

QwtPlotZoneItem

A horizontal or vertical section

QwtPlotTextLabel

In opposite to a QwtPlotMarker the text is not aligned to a plot coordinate but according to the geometry of the canvas (f.e top/centered for a title). See "playground/curvetracker".

2.2 Scales beyond linear and logarithmic transformations

QwtScaleTransformation has been replaced by QwtTransform and its derived classes:

- QwtTransform
- QwtNullTransform
- · QwtLogTransform
- QwtPowerTransform

Individual transformations (f.e. different scaling for special sections) can be implemented by overloading Qwt-Transform (see playground/scaleengine).

QwtLinearScaleEngine and QwtLogScaleEngine are not limited to base 10 anymore.

2.2.1 Datetime scales

A set of a new classes for displaying datetime values:

QwtDate

A collection of methods to convert between QDateTime and doubles

QwtDateScaleEngine

A scale engine that aligns and finds ticks in terms of datetime units.

QwtDateScaleDraw

A scale draw mapping values to datetime strings.

Scales for Qt::UTC and Qt::LocalTime are supported.

2.3 Redesign of the dial and meter widgets

Many parts of the class design of the dial and meter widgets were left over from the 90s (Qwt 0.2, Qt 1.1).

The derivation tree is simpler and more logical:

- QwtAbstractScale is a QWidget
- QwtAbstractSlider is a QwtAbstractScale. (for sliders without scales QAbstractSlider should be the base class)
- · QwtThermo is also a QwtAbstractScale
- · QwtDial, QwtKnob, QwtSlider are derived from QwtAbstractSlider
- · QwtCounter is derived from QWidget

QwtDoubleRange has been removed.

All classes use the terminology known from QAbstractSlider - as far as possible. The extended system for scales is completely supported.

2.4 Basic support for an OpenGL plot canvas

QwtPlotGLCanvas offers the option to draw plot items using an OpenGL paint engine (QPaintEngine::OpenG-L/OpenGL2), This is not what could be implemented with native OpenGL, but it offers hardware acceleration in environments, where the raster paint engine is the only option. (f.e Qt4/Windows, or Qt5 on all platforms).

QwtPlotGLCanvas is in an experimental state and is not recommended for average use cases.

2.5 A new system for plot legends

QwtLegend has been decoupled from QwtPlot and can be replaced by application specific implementations. Plot items and the legend exchange the information using QwtLegendData.

QwtPlotLegendItem is a new plot item that displays a legend on the plot canvas.

The following examples demonstrate how to use the new system:

- examples/legends shows how to use the new legend system
- examples/stockchart implementats a QTreeView with checkable items as legend

2.6 Off-screen paint device for vector graphics

QwtGraphic can be copied like QImage or QPixmap but is scalable like QSvgGenerator. It is implemented as a record/replay paint device like QPicture.

2.7 QwtWidgetOverlay

QwtWidgetOverlay is a base class for implementing widget overlays - primarily used for use cases like graphical editors or running cursors for the plot canvas.

The following examples show how to use overlays:

- · examples/itemeditor
- examples/curvetracker

QwtPicker (-> QwtPlotPicker, QwtPlotZoomer) internally uses QwtWidgetOverlay now, making it easier to implement individual rubber bands.

2.8 QwtSymbol

New symbol types have been introduced:

- · QwtSymbol::Path
- QwtSymbol::Pixmap
- QwtSymbol::Graphic
- QwtSymbol::SvgDocument

QwtSymbol autodetect the most performant paint strategy for a paint device what is in most situations using a QPixmap cache.

QwtSymbol::setPinPoint() allows to align the symbol individually, f.e to the position of the peak of an arrow.

2.9 QwtPlotCurve

Some optimizations that got lost with introducing the floating point based render code with Qwt 6.0 have been reenabled. Other specific optimizations have been added.

New paint attributes:

- · QwtPlotCurve::FilterPoints
- QwtPlotCurve::MinimizeMemory
- QwtPlotCurve::ImageBuffer

QwtPlotCurve::CacheSymbols has been removed, as caching is implemented in QwtSymbol now.

QwtPlotCurve::drawLines(), QwtPlotCurve::drawDots(), QwtPlotCurve::drawSteps() and QwtPlotCurve::draw-Sticks() are virtual now.

2.10 QwtPlot

A footer similar to a title has been added.

QwtPlot::ExternalLegend is obsolete with the new system for legends. The signals QwtPlot::legendClicked(), Qwt-Plot::legendChecked() have been removed. Applications need to connect to QwtLegend::clicked() and QwtLegend::checked().

To support using an OpenGL canvas QwtPlot::setCanvas has been added. This has 2 important implications for the application code:

- QwtPlot::canvas() returns QWidget and needs to be casted, when using methods of QwtPlotCanvas.
- QwtPlotCanvas can be created and assigned in application code, what makes it possible to derive and overload methods.

The initialization of a plot canvas with Qwt 6.1 will probably look like this:

```
QwtPlotCanvas* canvas = new QwtPlotCanvas();
canvas->setXY( ... );
...
plot->setCanvas( canvas );
```

To have a consistent API QwtPlot::setPlotLayout() has been added,

2.11 Other

2.11.1 QwtScaleDiv

The following methods have been added:

- QwtScaleDiv::inverted()
- QwtScaleDiv::bounded()
- QwtScaleDiv::isEmpty()
- QwtScaleDiv::isIncreasing()
- · QDebug operator

The following methods have been removed:

QwtScaleDiv::isValid(), QwtScaleDiv::invalidate()
 The valid state was left over from early Qwt versions indicating a state of the autoscaler.

2.11 Other 7

2.11.2 QwtScaleEngine

The following methods have been added:

- QwtScaleEngine::setBase()
- QwtScaleEngine::setTransformation()

2.11.3 QwtPlotLayout

The following flags have been added:

- QwtPlotLayout::IgnoreTitle
- QwtPlotLayout::IgnoreFooter
- QwtPlotLayout::setAlignCanvasToScale()

2.11.4 QwtPlotCanvas

Rounded borders (like with style sheets) can configured using QwtPlotCanvas::setBorderRadius();

2.11.5 Other changes

- QwtWeedingCurveFitter
 - QwtWeedingCurveFitter::setChunkSize() has been added, with drastic performance improvements for huge sets of points.
- QwtPlotRenderer The frame of the plot canvas can be rendered, what makes the result even closer to WYS-WYG. QwtPlotRenderer::exportTo() has been added.
- QwtSystemClock For Qt >= 4.9 QwtSystemClock uses QElapsedTimer internally. As it doesn't support a similar feature, QwtSystemClock::precision() has been removed.
- QwtPlotAbstractSeriesItem

QwtPlotAbstractSeriesItem has been split into QwtPlotSeriesItem and QwtPlotAbstractSeriesStore.

QwtText

A metatype declaration has been added, so that QwtText can be used with QVariant.

· QwtEventPattern, QwtPanner, QwtMagnifier

Forgotten Qt3 leftovers have been fixed: int -> Qt::KeyboardModifiers

- QPen Qt5/Qt4 incompatibility The default pen width for Qt5 is 1, what makes it a non cosmetic. To hide
 this nasty incompatibility several setPen() methods have been added the build pens with a width 0. See
 QPen::isCosmetic(),
- qwtUpperSampleIndex()

A binary search algorithm for sorted samples

- QwtMatrixRasterData QwtMatrixRasterData::setValue() has been added
- QwtPicker QwtPicker::rubberBandWidget(), QwtPicker::trackerWidget() have been replaced by QwtPicker::rubberBandOverlay(), QwtPicker::trackerOverlay(). QwtPicker::rubberBandMask() has been added. Qwt-Picker::pickRect() has been replaced by QwtPicker::pickArea()
- QwtPlotItem QwtPlotItem::ItemInterest has been added. QwtPlotItem::setRenderThreadCount() was shifted from QwtPlotRasterItem.

• ...

2.12 Summary of the new classes

- QwtAbstractLegend
- QwtDate
- QwtDateScaleDraw
- QwtDateScaleEngine
- QwtGraphic
- QwtLegendData
- QwtLegendLabel
- QwtPainterCommand
- QwtPixelMatrix
- QwtPlotAbstractBarChart
- QwtPlotBarChart
- QwtPlotMultiBarChart
- QwtPlotGLCanvas
- QwtPlotLegendItem
- QwtPlotShapeItem
- QwtPlotTextLabel
- QwtPlotTradingCurve
- QwtPlotZoneItem
- · QwtPointData
- QwtPointMapper
- QwtTransform, QwtNullTransform, QwtLogTransform, QwtPowerTransform
- QwtWidgetOverlay

3 Installing Qwt

3.1 Download

Stable Qwt releases are available from the Qwt project page.

Qwt-6.1.1 consists of 4 files:

• qwt-6.1.1.zip

Zip file with the Qwt sources and the html documentation for Windows

• qwt-6.1.1.tar.bz2

Compressed tar file with the Qwt sources and the html documentation for UNIX systems (Linux, Mac, ...)

• qwt-6.1.1.pdf

Qwt documentation as PDF document.

3.2 Installing Qwt 9

qwt-6.1.1.qch

Qwt documentation as Qt Compressed Help document, that can be loaded into the Qt Assistant or Creator. In the Qt Creator context sensitive help will be available like for Qt classes.

Precompiled Qwt Designer plugins, that are compatible with some binary packages of the Qt Creator:

qwtdesigner-6.1.1-*.zip

3.2 Installing Qwt

Beside headers, libraries and the html version of the class documentation a proper Qwt installation contains a Designer plugin and a Qwt features file for building applications using Qwt.

All files will be copied to an installation directory, that is configurable by editing qwtconfig.pri. Its default settings is:

Windows

C:\Qwt-6.1.1

· Unix like systems

/usr/local/qwt-6.1.1

For the rest of the document this install path will be written as \${QWT_ROOT}\$ and needs to be replaced by the real path in all commands below.

It is not unlikely, to have more than one installation of Qwt on the same system. F.e for using the Qwt Designer plugin in the Qt Creator a version of Qwt is necessary with the same Qt and compiler combination, that had been used for building the Qt Creator (see "Help->About Qt Creator ...").

Installing Qwt is done in 3 steps, that are quite common on UNIX systems.

1. Configuration

In the configuration step all parameters are set to control how to build and install Qwt

2. Build

In the build step binaries are built from the source files.

3. Installation

The installation copies and rearranges all files that are necessary to build Qwt applications to a target directory.

The installation doesn't modify the system beside copying files to a directory in a proper way. After removing build and installation directories the system is in the same state as it was before.

3.2.1 Configuration

Configuring Qwt has to be done by editing the Project files used for building:

- qwtbuild.pri
 - qwtbuild.pri contains settings for how to build Qwt. All settings of this file are only for building Qwt itself and doesn't have an impact on how an application using Qwt is built. Usually its default settings doesn't need to be modified.
- · qwtconfig.pri

qwtconfig.pri defines what modules of Qwt will be built and where to install them. qwtconfig.pri gets installed together with the Qwt features file qwt.prf and all its settings are known to project files for building Qwt applications.

In qwtconfig.pri the meaning of each option is explained in detail - it's worth reading it before running into problems later.

3.2.2 Build and installation

The Qt Creator is a graphical frontend for calling qmake/make and - technically - it could be used for building and installing Qwt. But as this way requires a lot more understanding of details the following step by step instructions are for the easier way using the command line.

3.2.2.1 Unix-like systems

The first step before creating the Makefile is to check that the correct version of qmake is used. F.e. on older Linux distribution you often find a Qt3 qmake and in the path.

The default setting of qmake is to generate a makefile that builds Qwt for the same environment where the version of qmake has been built for. So creating a makefile usually means something like:

```
cd qwt-6.1.1
/usr/local/Qt-5.0.1/bin/qmake qwt.pro
```

The generated Makefile includes all paths related to the chosen Qt version and the next step is:

make

(On multicore systems you can speed up building the Qwt libraries with running several jobs simultaneously: f.e. "make -j4" on a dual core.)

Finally you have to install everything below the directories you have specified in qwtconfig.pri. Usually this is one of the system directories (/usr/local, /opt, ...) where you don't have write permission and then the installation needs to be done as root:

```
sudo make install
```

(On systems where sudo is not supported you can do the same with: su -c "make install")

3.2.2.2 Windows

Qt packages offer a command line interface, that can be found in the Qt application menu: f.e "All Programs -> Qt -> Command Prompt". It is not mandatory to use it, but probably the easiest way as it offers an environment, where everything is initialized for a version of Qt (f.e qmake is in the PATH).

Creating a makefile usually means something like:

```
cd qwt-6.1.1 qmake qwt.pro
```

The generated makefile includes all paths related to the chosen Qt version.

3.2.2.2.1 MinGW

For MinGW builds the name of the make tool is "mingw32-make"

```
mingw32-make
```

(On multicore systems you can speed up building the Qwt libraries with running several jobs simultaneously: "mingw32-make -j")

Finally you have to install everything below the directories you have specified in qwtconfig.pri.

```
mingw32-make install
```

3.2.2.2.2 MSVC

For MSVC builds the name of the make tool is "nmake". Alternatively it is possible to use "jom" (http-://qt-project.org/wiki/jom), that is usually included in a Qt Creator package.

nmake

Finally you have to install everything below the directories you have specified in qwtconfig.pri.

nmake install

3.3 Qwt and the Qt tool chain

3.3.1 Designer plugin

The Designer plugin and the corresponding Qwt library (if the plugin has not been built self containing) have to be compatible with Qt version of the application loading it (usually the Qt Creator) - what is often a different version of the Qt libraries you want to build your application with. F.e on Windows the Qt Creator is usually built with a MSVC compiler - even if included in a MinGW package!

To help Qt Designer/Creator with locating the Qwt Designer plugin you have to set the environment variable QT_P-LUGIN_PATH, modify qt.conf - or install the plugin to one of the application default paths.

The Qt documentation explains all options in detail:

- http://qt-project.org/doc/qt-5.0/qtdoc/deployment-plugins.html
- http://qt-project.org/doc/qtcreator-2.7/adding-plugins.html.

F.e. on a Linux system you could add the following lines to .bashrc:

```
QT_PLUGIN_PATH="${QWT_ROOT}/plugins:$QT_PLUGIN_PATH"
export QT_PLUGIN_PATH
```

When the plugin has not been built including the Qwt library (see QwtDesignerSelfContained in qwtconfig.pri) the Qt Designer/Creator also needs to locate the Qwt libraries. On Unix systems the path to the installed library is compiled into the plugin (see rpath, ldd), but on Windows the Qt Creator needs to be configured ((Running a Qwt application) in the same way as for any application using Qwt.

In case of problems the diagnostics of Qt Creator and Designer are very limited (usually none), but setting the environment variable QT_DEBUG_PLUGINS might help. In the Qt Creator it is possible to check which plugins were loaded successfully and for certain problems it also lists those that were recognized but failed (Tools > Form Editor > About Qt Designer Plugins).

3.3.2 Online Help

The Qwt class documentation can be loaded into the Qt Creator:

- open the settings dialog from the Tools->Options menu
- raise the tab "Help->Documentation".
- press the Add button and select qwt-6.1.1.qch.

Now the context sensitive help (F1) works for Qwt classes.

For browsing the documentation in the Qt Assistant:

- open the settings dialog from the Edit->Preferences menu
- raise the tab Documentation.
- press the Add button and select qwt-6.1.1.qch.

3.4 Building a Qwt application

All flags and settings that are necessary to compile and link an application using Qwt can be found in the file \${QWT_ROOT}/features/qwt.prf.

When using qmake it can included from the application project file in 2 different ways:

· Adding Qwt as qmake feature

When using the qmake feature mechanism you can bind a special version of qmake to a special installation of Qwt without having to add this dependency to the application project. How to add Qwt as feature is documented in the qmake docs.

After adding Qwt as a feature f.e on Linux as a persistent property

```
qmake -set QMAKEFEATURES ${QWT_ROOT}/features
```

.. the following line can be added to the application project file:

```
CONFIG += qwt
```

· Including gwt.prf in the application project file

Instead of using qwt.prf as qmake feature it can be included from the application project file:

```
include ( ${QWT_ROOT}/features/qwt.prf )
```

The advantage of using a direct include is, that all settings of qwt.prf are known to the application project file (qmake features are included after the application project file has been parsed) and it can be implemented depending on - f.e. settings made in qwtconfig.pri.

On Unix platforms it is possible to link a runtime path into the executable, so that the location of the Qwt libraries can be found without having to configure a runtime environment:

- QMAKE_LFLAGS_RPATH
- QMAKE_RPATH
- QMAKE_RPATHDIR

3.5 Running a Qwt application

When using Qwt as shared library (DLL) the dynamic linker has to find it according to the rules of the operating system.

3.5.1 Windows

The only reasonable way to configure the runtime environment - without having to copy the Qwt libraries around - is to modify the PATH variable. F.e. this could be done by adding the following line to some batch file:

```
set PATH=%PATH%;${QWT}_{ROOT}\
```

3.5.2 GNU/Linux

Read the documentation about:

- Idconfig
- · /etc/ld.so.conf
- LD LIBRARY PATH

Using the *Idd* command a configuration can be tested.

4 Qwt License, Version 1.0

Qwt License Version 1.0, January 1, 2003

The Qwt library and included programs are provided under the terms of the GNU LESSER GENERAL PUBLIC LICENSE (LGPL) with the following exceptions:

- Widgets that are subclassed from Qwt widgets do not constitute a derivative work.
- 2. Static linking of applications and widgets to the Qwt library does not constitute a derivative work and does not require the author to provide source code for the application or widget, use the shared Qwt libraries, or link their applications or widgets against a user-supplied version of Qwt.

If you link the application or widget to a modified version of Qwt, then the changes to Qwt must be provided under the terms of the LGPL in sections $1,\ 2,\$ and 4.

3. You do not have to provide a copy of the Qwt license with programs that are linked to the Qwt library, nor do you have to identify the Qwt license in your program or documentation as required by section 6 of the LGPL.

However, programs must still identify their use of Qwt. The following example statement can be included in user documentation to satisfy this requirement:

[program/widget] is based in part on the work of the Qwt project (http://qwt.sf.net).

GNU LESSER GENERAL PUBLIC LICENSE Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages—typically libraries—of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling

it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

O. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated

straightforwardly into another language. (Hereinafter, translation is included without limitation in the term $\mbox{"modification".}$)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee. $\,$

- 2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
 - a) The modified work must itself be a software library.
 - b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
 - c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
 - d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

- 4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.
- If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.
- 5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

- a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)
- b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.
- c) Accompany the work with a written offer, valid for at

least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

- d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.
- e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

- 7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:
 - a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.
 - b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.
- 8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
- 9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.
- 10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.
- 11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the

integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

- 12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
- 13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

- 15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
- 16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

5 Curve Plots 19

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990 Ty Coon, President of Vice

That's all there is to it!

- 5 Curve Plots
- 6 Scatter Plot
- 7 Spectrogram, Contour Plot

/*!

- 8 Histogram
- 9 Dials, Compasses, Knobs, Wheels, Sliders, Thermos
- 10 Hierarchical Index
- 10.1 Class Hierarchy

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

| 3 1 |
|------------|
| 31 |
| 259 |
| 32 |
| 176 |
| 264 |
| 296 |
| 603 |
| |

| 186 |
|-----|
| 321 |
| 128 |
| 200 |
| 381 |
| 234 |
| 400 |
| 477 |
| 318 |
| 412 |
| 420 |
| 214 |
| 139 |
| 69 |
| 511 |
| 34 |
| 52 |
| 118 |
| 63 |
| 79 |
| 166 |
| 562 |
| 607 |
| 88 |
| 228 |
| 398 |
| 540 |
| 624 |
| |

| QwtWidgetOverlay | 637 |
|--------------------------------------|-----|
| QwtAbstractScaleDraw | 42 |
| QwtRoundScaleDraw | 507 |
| QwtCompassScaleDraw | 85 |
| QwtScaleDraw | 520 |
| QwtDateScaleDraw | 105 |
| QwtAbstractSeriesStore | 50 |
| QwtPlotSeriesItem | 435 |
| QwtPlotAbstractBarChart | 285 |
| QwtPlotBarChart | 290 |
| QwtPlotMultiBarChart | 390 |
| QwtPlotCurve | 302 |
| QwtPlotHistogram | 332 |
| QwtPlotIntervalCurve | 341 |
| QwtPlotSpectroCurve | 444 |
| QwtPlotTradingCurve | 464 |
| QwtSeriesStore < T > | 552 |
| QwtSeriesStore < QPointF > | 552 |
| QwtPlotBarChart | 290 |
| QwtPlotCurve | 302 |
| QwtSeriesStore < QwtIntervalSample > | 552 |
| QwtPlotHistogram | 332 |
| QwtPlotIntervalCurve | 341 |
| QwtSeriesStore < QwtOHLCSample > | 552 |
| QwtPlotTradingCurve | 464 |
| QwtSeriesStore < QwtPoint3D > | 552 |
| QwtPlotSpectroCurve | 444 |
| QwtSeriesStore < QwtSetSample > | 552 |
| QwtPlotMultiBarChart | 390 |
| QwtClipper | 71 |
| QwtColorMap | 72 |
| QwtAlphaColorMap | 61 |

| QwtLinearColorMap | 189 |
|----------------------------|-----|
| QwtColumnRect | 75 |
| QwtColumnSymbol | 76 |
| QwtCompassRose | 84 |
| QwtSimpleCompassRose | 559 |
| QwtCurveFitter | 99 |
| QwtSplineCurveFitter | 574 |
| QwtWeedingCurveFitter | 622 |
| QwtDate | 100 |
| QwtDialNeedle | 124 |
| QwtCompassMagnetNeedle | 82 |
| QwtCompassWindArrow | 87 |
| QwtDialSimpleNeedle | 126 |
| QwtEventPattern | 134 |
| QwtPicker | 234 |
| QwtInterval | 152 |
| QwtIntervalSample | 160 |
| QwtIntervalSymbol | 163 |
| QwtLegendData | 183 |
| QwtOHLCSample | 218 |
| QwtPainter | 220 |
| QwtPainterCommand | 225 |
| QwtPickerMachine | 256 |
| QwtPickerClickPointMachine | 252 |
| QwtPickerClickRectMachine | 252 |
| QwtPickerDragLineMachine | 254 |
| QwtPickerDragPointMachine | 254 |
| QwtPickerDragRectMachine | 255 |
| QwtPickerPolygonMachine | 258 |
| QwtPickerTrackerMachine | 259 |
| QwtPlotDict | 316 |
| QwtPlot | 264 |

| QwtPlotItem | 348 | | |
|--------------------------------|-----|--|--|
| QwtPlotGrid | 327 | | |
| QwtPlotLegendItem | | | |
| QwtPlotMarker | | | |
| QwtPlotRasterItem | 407 | | |
| QwtPlotSpectrogram | 448 | | |
| QwtPlotScaleItem | 429 | | |
| QwtPlotSeriesItem | 435 | | |
| QwtPlotShapeItem | 437 | | |
| QwtPlotSvgltem | 457 | | |
| QwtPlotTextLabel | 460 | | |
| QwtPlotZoneItem | 474 | | |
| QwtPlotLayout | 363 | | |
| QwtPoint3D | 486 | | |
| QwtPointMapper | | | |
| QwtPointPolar | 496 | | |
| QwtRasterData | 502 | | |
| QwtMatrixRasterData | 210 | | |
| QwtScaleArithmetic | 513 | | |
| QwtScaleDiv | 514 | | |
| QwtScaleEngine | 530 | | |
| QwtLinearScaleEngine | 193 | | |
| QwtDateScaleEngine | 110 | | |
| QwtLogScaleEngine | 196 | | |
| QwtScaleMap | 536 | | |
| QwtSeriesData< T > | 550 | | |
| QwtArraySeriesData< T > | 67 | | |
| QwtSeriesData < QPointF > | 550 | | |
| QwtArraySeriesData < QPointF > | 67 | | |
| QwtPointSeriesData | 499 | | |
| QwtCPointerData | 97 | | |
| QwtPointArrayData | 489 | | |

| QwtSyntheticPointData | 587 |
|---|-----|
| QwtSeriesData < QwtIntervalSample > | 550 |
| QwtArraySeriesData< QwtIntervalSample > | 67 |
| QwtIntervalSeriesData | 161 |
| QwtSeriesData < QwtOHLCSample > | 550 |
| QwtArraySeriesData < QwtOHLCSample > | 67 |
| QwtTradingChartData | 619 |
| QwtSeriesData< QwtPoint3D > | 550 |
| QwtArraySeriesData < QwtPoint3D > | 67 |
| QwtPoint3DSeriesData | 488 |
| QwtSeriesData < QwtSetSample > | 550 |
| QwtArraySeriesData < QwtSetSample > | 67 |
| QwtSetSeriesData | 557 |
| QwtSetSample | 555 |
| QwtSpline | 570 |
| QwtSymbol | 576 |
| QwtSystemClock | 591 |
| QwtText | 592 |
| QwtTextEngine | 601 |
| QwtMathMLTextEngine | 208 |
| QwtPlainTextEngine | 261 |
| QwtRichTextEngine | 505 |
| QwtTransform | 620 |
| QwtLogTransform | 199 |
| QwtNullTransform | 217 |
| QwtPowerTransform | 500 |
| | |
| Class Index | |

11

11.1 Class List

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

QwtEventPattern::KeyPattern

A pattern for key events **31** 11.1 Class List 25

| QwtEventPattern::MousePattern A pattern for mouse events | 31 |
|--|----|
| QwtAbstractLegend Abstract base class for legend widgets | 32 |
| QwtAbstractScale An abstract base class for widgets having a scale | 34 |
| QwtAbstractScaleDraw A abstract base class for drawing scales | 42 |
| QwtAbstractSeriesStore Bridge between QwtSeriesStore and QwtPlotSeriesItem | 50 |
| QwtAbstractSlider An abstract base class for slider widgets with a scale | 52 |
| QwtAlphaColorMap QwtAlphaColorMap varies the alpha value of a color | 61 |
| QwtAnalogClock An analog clock | 63 |
| QwtArraySeriesData< T > Template class for data, that is organized as QVector | 67 |
| QwtArrowButton Arrow Button | 69 |
| QwtClipper Some clipping algorithms | 71 |
| QwtColorMap is used to map values into colors | 72 |
| QwtColumnRect Directed rectangle representing bounding rectangle and orientation of a column | 75 |
| QwtColumnSymbol A drawing primitive for columns | 76 |
| QwtCompass A Compass Widget | 79 |
| QwtCompassMagnetNeedle A magnet needle for compass widgets | 82 |
| QwtCompassRose Abstract base class for a compass rose | 84 |
| QwtCompassScaleDraw A special scale draw made for QwtCompass | 85 |
| QwtCompassWindArrow An indicator for the wind direction | 87 |
| QwtCounter The Counter Widget | 88 |
| QwtCPointerData Data class containing two pointers to memory blocks of doubles | 97 |

| QwtCurveFitter Abstract base class for a curve fitter | 99 |
|--|-----|
| QwtDate A collection of methods around date/time values | 100 |
| QwtDateScaleDraw A class for drawing datetime scales | 105 |
| QwtDateScaleEngine A scale engine for date/time values | 110 |
| QwtDial class provides a rounded range control | 115 |
| QwtDialNeedle Base class for needles that can be used in a QwtDial | 124 |
| QwtDialSimpleNeedle A needle for dial widgets | 126 |
| QwtDynGridLayout Lays out widgets in a grid, adjusting the number of columns and rows to the current size | 128 |
| QwtEventPattern A collection of event patterns | 134 |
| QwtGraphic A paint device for scalable graphics | 139 |
| QwtInterval A class representing an interval | 152 |
| QwtIntervalSample A sample of the types (x1-x2, y) or (x, y1-y2) | 160 |
| QwtIntervalSeriesData Interface for iterating over an array of intervals | 161 |
| QwtIntervalSymbol A drawing primitive for displaying an interval like an error bar | 163 |
| QwtKnob The Knob Widget | 166 |
| QwtLegend The legend widget | 176 |
| QwtLegendData Attributes of an entry on a legend | 183 |
| QwtLegendLabel A widget representing something on a QwtLegend | 186 |
| QwtLinearColorMap QwtLinearColorMap builds a color map from color stops | 189 |
| QwtLinearScaleEngine A scale engine for linear scales | 193 |
| QwtLogScaleEngine A scale engine for logarithmic scales | 196 |

11.1 Class List 27

| QwtLogTransform Logarithmic transformation | 199 |
|---|-----|
| QwtMagnifier QwtMagnifier provides zooming, by magnifying in steps | 200 |
| QwtMathMLTextEngine Text Engine for the MathML renderer of the Qt solutions package | 208 |
| QwtMatrixRasterData A class representing a matrix of values as raster data | 210 |
| QwtNullPaintDevice A null paint device doing nothing | 214 |
| QwtNullTransform Null transformation | 217 |
| QwtOHLCSample Open-High-Low-Close sample used in financial charts | 218 |
| QwtPainter A collection of QPainter workarounds | 220 |
| QwtPainterCommand | 225 |
| QwtPanner QwtPanner provides panning of a widget | 228 |
| QwtPicker QwtPicker provides selections on a widget | 234 |
| QwtPickerClickPointMachine A state machine for point selections | 252 |
| QwtPickerClickRectMachine A state machine for rectangle selections | 252 |
| QwtPickerDragLineMachine A state machine for line selections | 254 |
| QwtPickerDragPointMachine A state machine for point selections | 254 |
| QwtPickerDragRectMachine A state machine for rectangle selections | 255 |
| QwtPickerMachine A state machine for QwtPicker selections | 256 |
| QwtPickerPolygonMachine A state machine for polygon selections | 258 |
| QwtPickerTrackerMachine A state machine for indicating mouse movements | 259 |
| QwtPixelMatrix A bit field corresponding to the pixels of a rectangle | 259 |
| QwtPlainTextEngine A text engine for plain texts | 261 |

| QwtPlot A 2-D plotting widget | 264 |
|--|-----|
| QwtPlotAbstractBarChart Abstract base class for bar chart items | 285 |
| QwtPlotBarChart displays a series of a values as bars | 290 |
| QwtPlotCanvas Canvas of a QwtPlot | 296 |
| QwtPlotCurve A plot item, that represents a series of points | 302 |
| QwtPlotDict A dictionary for plot items | 316 |
| QwtPlotDirectPainter Painter object trying to paint incrementally | 318 |
| QwtPlotGLCanvas An alternative canvas for a QwtPlot derived from QGLWidget | 321 |
| QwtPlotGrid A class which draws a coordinate grid | 327 |
| QwtPlotHistogram QwtPlotHistogram represents a series of samples, where an interval is associated with a value ($y=f([x1,x2])$) | 332 |
| QwtPlotIntervalCurve QwtPlotIntervalCurve represents a series of samples, where each value is associated with an interval ($[y1,y2]=f(x)$) | 341 |
| QwtPlotItem Base class for items on the plot canvas | 348 |
| QwtPlotLayout Layout engine for QwtPlot | 363 |
| QwtPlotLegendItem A class which draws a legend inside the plot canvas | 372 |
| QwtPlotMagnifier QwtPlotMagnifier provides zooming, by magnifying in steps | 381 |
| QwtPlotMarker A class for drawing markers | 383 |
| QwtPlotMultiBarChart QwtPlotMultiBarChart displays a series of a samples that consist each of a set of values | 390 |
| QwtPlotPanner QwtPlotPanner provides panning of a plot canvas | 398 |
| QwtPlotPicker QwtPlotPicker provides selections on a plot canvas | 400 |
| QwtPlotRasterItem A class, which displays raster data | 407 |

11.1 Class List

| QwtPlotRenderer | |
|---|-----|
| Renderer for exporting a plot to a document, a printer or anything else, that is supported by QPainter/QPaintDevice | 412 |
| QwtPlotRescaler QwtPlotRescaler takes care of fixed aspect ratios for plot scales | 420 |
| QwtPlotScaleItem | |
| A class which draws a scale inside the plot canvas | 429 |
| QwtPlotSeriesItem Base class for plot items representing a series of samples | 435 |
| QwtPlotShapeItem A plot item, which displays any graphical shape, that can be defined by a QPainterPath | 437 |
| QwtPlotSpectroCurve Curve that displays 3D points as dots, where the z coordinate is mapped to a color | 444 |
| QwtPlotSpectrogram A plot item, which displays a spectrogram | 448 |
| QwtPlotSvgltem A plot item, which displays data in Scalable Vector Graphics (SVG) format | 457 |
| QwtPlotTextLabel A plot item, which displays a text label | 460 |
| QwtPlotTradingCurve QwtPlotTradingCurve illustrates movements in the price of a financial instrument over time | 464 |
| QwtPlotZoneItem A plot item, which displays a zone | 474 |
| QwtPlotZoomer QwtPlotZoomer provides stacked zooming for a plot widget | 477 |
| QwtPoint3D QwtPoint3D class defines a 3D point in double coordinates | 486 |
| QwtPoint3DSeriesData Interface for iterating over an array of 3D points | 488 |
| QwtPointArrayData Interface for iterating over two QVector <double> objects</double> | 489 |
| QwtPointMapper A helper class for translating a series of points | 491 |
| QwtPointPolar A point in polar coordinates | 496 |
| QwtPointSeriesData Interface for iterating over an array of points | 499 |
| QwtPowerTransform A transformation using pow() | 500 |
| QwtRasterData QwtRasterData defines an interface to any type of raster data | 502 |

| QwtRichTextEngine A text engine for Qt rich texts | 505 |
|---|-----|
| QwtRoundScaleDraw A class for drawing round scales | 507 |
| QwtSamplingThread A thread collecting samples at regular intervals | 511 |
| QwtScaleArithmetic Arithmetic including a tolerance | 513 |
| QwtScaleDiv A class representing a scale division | 514 |
| QwtScaleDraw A class for drawing scales | 520 |
| QwtScaleEngine Base class for scale engines | 530 |
| QwtScaleMap A scale map | 536 |
| QwtScaleWidget A Widget which contains a scale | 540 |
| QwtSeriesData < T > Abstract interface for iterating over samples | 550 |
| QwtSeriesStore < T > Class storing a QwtSeriesData object | 552 |
| QwtSetSample A sample of the types (x1xn, y) or (x, y1yn) | 555 |
| QwtSetSeriesData Interface for iterating over an array of samples | 557 |
| QwtSimpleCompassRose A simple rose for QwtCompass | 559 |
| QwtSlider The Slider Widget | 562 |
| QwtSpline A class for spline interpolation | 570 |
| QwtSplineCurveFitter A curve fitter using cubic splines | 574 |
| QwtSymbol A class for drawing symbols | 576 |
| QwtSyntheticPointData Synthetic point data | 587 |
| QwtSystemClock QwtSystemClock provides high resolution clock time functions | 591 |
| QwtText A class representing a text | 592 |

12 Class Documentation 31

| QwtTextEngine | |
|---|-----|
| Abstract base class for rendering text strings | 601 |
| QwtTextLabel | |
| A Widget which displays a QwtText | 603 |
| QwtThermo | |
| The Thermometer Widget | 607 |
| QwtTradingChartData | 619 |
| QwtTransform | |
| A transformation between coordinate systems | 620 |
| QwtWeedingCurveFitter | |
| A curve fitter implementing Douglas and Peucker algorithm | 622 |
| QwtWheel | |
| The Wheel Widget | 624 |
| QwtWidgetOverlay | |
| An overlay for a widget | 637 |
| | |

12 Class Documentation

12.1 QwtEventPattern::KeyPattern Class Reference

A pattern for key events.

```
#include <qwt_event_pattern.h>
```

Public Member Functions

KeyPattern (int keyCode=Qt::Key_unknown, Qt::KeyboardModifiers modifierCodes=Qt::NoModifier)
 Constructor.

Public Attributes

int key

Key code.

• Qt::KeyboardModifiers modifiers

Modifiers.

12.1.1 Detailed Description

A pattern for key events.

12.2 QwtEventPattern::MousePattern Class Reference

A pattern for mouse events.

```
#include <qwt_event_pattern.h>
```

Public Member Functions

MousePattern (Qt::MouseButton btn=Qt::NoButton, Qt::KeyboardModifiers modifierCodes=Qt::NoModifier)
 Constructor.

Public Attributes

 Qt::MouseButton button Button.

Qt::KeyboardModifiers modifiers
 Keyboard modifier.

12.2.1 Detailed Description

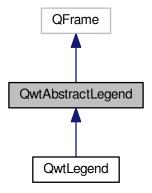
A pattern for mouse events.

12.3 QwtAbstractLegend Class Reference

Abstract base class for legend widgets.

#include <qwt_abstract_legend.h>

Inheritance diagram for QwtAbstractLegend:



Public Slots

virtual void updateLegend (const QVariant &itemInfo, const QList< QwtLegendData > &data)=0
 Update the entries for a plot item.

Public Member Functions

- QwtAbstractLegend (QWidget *parent=NULL)
- virtual ~QwtAbstractLegend ()

Destructor

• virtual void renderLegend (QPainter *painter, const QRectF &rect, bool fillBackground) const =0

- virtual bool isEmpty () const =0
- · virtual int scrollExtent (Qt::Orientation) const

12.3.1 Detailed Description

Abstract base class for legend widgets.

Legends, that need to be under control of the QwtPlot layout system need to be derived from QwtAbstractLegend.

Note

Other type of legends can be implemented by connecting to the QwtPlot::legendDataChanged() signal. But as these legends are unknown to the plot layout system the layout code (on screen and for QwtPlotRenderer) need to be organized in application code.

See Also

QwtLegend

12.3.2 Constructor & Destructor Documentation

12.3.2.1 QwtAbstractLegend::QwtAbstractLegend (QWidget * parent = NULL) [explicit]

Constructor

Parameters

| parent | Parent widget |
|--------|---------------|

12.3.3 Member Function Documentation

12.3.3.1 virtual bool QwtAbstractLegend::isEmpty() const [pure virtual]

Returns

True, when no plot item is inserted

Implemented in QwtLegend.

12.3.3.2 virtual void QwtAbstractLegend::renderLegend (QPainter * painter, const QRectF & rect, bool fillBackground) const [pure virtual]

Render the legend into a given rectangle.

Parameters

| pa | ainter | Painter |
|------------|--------|---|
| | rect | Bounding rectangle |
| fillBackgr | ound | When true, fill rect with the widget background |

See Also

renderLegend() is used by QwtPlotRenderer

Implemented in QwtLegend.

12.3.3.3 int QwtAbstractLegend::scrollExtent (Qt::Orientation orientation) const [virtual]

Return the extent, that is needed for elements to scroll the legend (usually scrollbars),

Parameters

| orientation | Orientation |
|-------------|-------------|
|-------------|-------------|

Returns

Extent of the corresponding scroll element

Reimplemented in QwtLegend.

12.3.3.4 virtual void QwtAbstractLegend::updateLegend (const QVariant & itemInfo, const QList< QwtLegendData > & data) [pure virtual], [slot]

Update the entries for a plot item.

Parameters

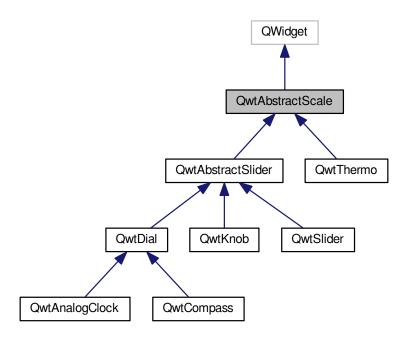
| itemInfo | Info about an item |
|----------|--|
| data | List of legend entry attributes for the item |

12.4 QwtAbstractScale Class Reference

An abstract base class for widgets having a scale.

#include <qwt_abstract_scale.h>

Inheritance diagram for QwtAbstractScale:



Public Member Functions

QwtAbstractScale (QWidget *parent=NULL)

virtual ~QwtAbstractScale ()

Destructor.

void setScale (double lowerBound, double upperBound)

Specify a scale.

void setScale (const QwtInterval &)

Specify a scale.

void setScale (const QwtScaleDiv &)

Specify a scale.

- · const QwtScaleDiv & scaleDiv () const
- void setLowerBound (double value)
- double lowerBound () const
- void setUpperBound (double value)
- · double upperBound () const
- void setScaleStepSize (double stepSize)

Set the step size used for calculating a scale division.

- double scaleStepSize () const
- void setScaleMaxMajor (int ticks)

Set the maximum number of major tick intervals.

- int scaleMaxMinor () const
- void setScaleMaxMinor (int ticks)

Set the maximum number of minor tick intervals.

- int scaleMaxMajor () const
- void setScaleEngine (QwtScaleEngine *)

Set a scale engine.

- const QwtScaleEngine * scaleEngine () const
- QwtScaleEngine * scaleEngine ()
- int transform (double) const
- double invTransform (int) const
- · bool isInverted () const
- double minimum () const
- double maximum () const
- const QwtScaleMap & scaleMap () const

Protected Member Functions

- void rescale (double lowerBound, double upperBound, double stepSize)
- void setAbstractScaleDraw (QwtAbstractScaleDraw *)

Set a scale draw.

- const QwtAbstractScaleDraw * abstractScaleDraw () const
- QwtAbstractScaleDraw * abstractScaleDraw ()
- virtual void scaleChange ()

Notify changed scale.

12.4.1 Detailed Description

An abstract base class for widgets having a scale.

The scale of an QwtAbstractScale is determined by a QwtScaleDiv definition, that contains the boundaries and the ticks of the scale. The scale is painted using a QwtScaleDraw object.

The scale division might be assigned explicitly - but usually it is calculated from the boundaries using a QwtScale-Engine.

The scale engine also decides the type of transformation of the scale (linear, logarithmic ...).

- 12.4.2 Constructor & Destructor Documentation
- 12.4.2.1 QwtAbstractScale::QwtAbstractScale (QWidget * parent = \mathtt{NULL})

Constructor

Parameters

parent Parent widget

 $Creates\ a\ default\ {\color{blue}QwtScaleDraw}\ and\ a\ {\color{blue}QwtLinearScaleEngine}.\ The\ initial\ scale\ boundaries\ are\ set\ to\ [\ 0.0,\ 100.0\]$

The scaleStepSize() is initialized to 0.0, scaleMaxMajor() to 5 and scaleMaxMajor to 3.

12.4.3 Member Function Documentation

12.4.3.1 const QwtAbstractScaleDraw * QwtAbstractScale::abstractScaleDraw () const [protected]

Returns

Scale draw

See Also

setAbstractScaleDraw()

12.4.3.2 QwtAbstractScaleDraw * QwtAbstractScale::abstractScaleDraw() [protected]

Returns

Scale draw

See Also

setAbstractScaleDraw()

12.4.3.3 double QwtAbstractScale::invTransform (int value) const

Translate a widget coordinate into a scale value

Parameters

value | Widget coordinate

Returns

Corresponding scale coordinate for value

See Also

scaleMap(), transform()

12.4.3.4 bool QwtAbstractScale::isInverted () const

Returns

True, when the scale is increasing in opposite direction to the widget coordinates

12.4.3.5 double QwtAbstractScale::lowerBound () const

Returns

Lower bound of the scale

See Also

setLowerBound(), setScale(), upperBound()

```
12.4.3.6 double QwtAbstractScale::maximum ( ) const
```

Returns

The boundary with the larger value

See Also

```
minimum(), lowerBound(), upperBound()
```

12.4.3.7 double QwtAbstractScale::minimum () const

Returns

The boundary with the smaller value

See Also

```
maximum(), lowerBound(), upperBound()
```

12.4.3.8 void QwtAbstractScale::rescale (double lowerBound, double upperBound, double stepSize) [protected]

Recalculate the scale division and update the scale.

Parameters

| lowerBound | Lower limit of the scale interval |
|------------|-----------------------------------|
| upperBound | Upper limit of the scale interval |
| stepSize | Major step size |

See Also

```
scaleChange()
```

12.4.3.9 const QwtScaleDiv & QwtAbstractScale::scaleDiv () const

Returns

Scale boundaries and positions of the ticks

The scale division might have been assigned explicitly or calculated implicitly by rescale().

12.4.3.10 const QwtScaleEngine * QwtAbstractScale::scaleEngine () const

Returns

Scale engine

See Also

setScaleEngine()

 $12.4.3.11 \quad \textbf{QwtScaleEngine} * \textbf{QwtAbstractScale::scaleEngine} \left(\begin{array}{c} \\ \end{array} \right)$

Returns

Scale engine

See Also

setScaleEngine()

```
12.4.3.12 const QwtScaleMap & QwtAbstractScale::scaleMap ( ) const
Returns
      Map to translate between scale and widget coordinates
12.4.3.13 int QwtAbstractScale::scaleMaxMajor ( ) const
Returns
      Maximal number of major tick intervals
See Also
      setScaleMaxMajor(), scaleMaxMinor()
12.4.3.14 int QwtAbstractScale::scaleMaxMinor ( ) const
Returns
      Maximal number of minor tick intervals
See Also
      setScaleMaxMinor(), scaleMaxMajor()
12.4.3.15 double QwtAbstractScale::scaleStepSize ( ) const
Returns
      Hint for the step size of the scale
See Also
      setScaleStepSize(), QwtScaleEngine::divideScale()
12.4.3.16 void QwtAbstractScale::setAbstractScaleDraw ( QwtAbstractScaleDraw * scaleDraw ) [protected]
Set a scale draw.
scaleDraw has to be created with new and will be deleted in the destructor or the next call of setAbstractScaleDraw().
See Also
      abstractScaleDraw()
12.4.3.17 void QwtAbstractScale::setLowerBound ( double value )
Set the lower bound of the scale
Parameters
             value Lower bound
See Also
      lowerBound(), setScale(), setUpperBound()
Note
```

For inverted scales the lower bound is greater than the upper bound

12.4.3.18 void QwtAbstractScale::setScale (double lowerBound, double upperBound)

Specify a scale.

Define a scale by an interval

The ticks are calculated using scaleMaxMinor(), scaleMaxMajor() and scaleStepSize().

Parameters

| Γ | lowerBound | lower limit of the scale interval |
|---|------------|-----------------------------------|
| | upperBound | upper limit of the scale interval |

Note

For inverted scales the lower bound is greater than the upper bound

12.4.3.19 void QwtAbstractScale::setScale (const QwtInterval & interval)

Specify a scale.

Define a scale by an interval

The ticks are calculated using scaleMaxMinor(), scaleMaxMajor() and scaleStepSize().

Parameters

| in | nterval | Interval |
|----|---------|----------|

12.4.3.20 void QwtAbstractScale::setScale (const QwtScaleDiv & scaleDiv)

Specify a scale.

scaleMaxMinor(), scaleMaxMajor() and scaleStepSize() and have no effect.

Parameters

| scaleDiv | Scale division |
|----------|----------------|

See Also

setAutoScale()

12.4.3.21 void QwtAbstractScale::setScaleEngine (QwtScaleEngine * scaleEngine)

Set a scale engine.

The scale engine is responsible for calculating the scale division and provides a transformation between scale and widget coordinates.

scaleEngine has to be created with new and will be deleted in the destructor or the next call of setScaleEngine.

12.4.3.22 void QwtAbstractScale::setScaleMaxMajor (int ticks)

Set the maximum number of major tick intervals.

The scale's major ticks are calculated automatically such that the number of major intervals does not exceed ticks.

The default value is 5.

Parameters

ticks Maximal number of major ticks.

See Also

scaleMaxMajor(), setScaleMaxMinor(), setScaleStepSize(), QwtScaleEngine::divideInterval()

12.4.3.23 void QwtAbstractScale::setScaleMaxMinor (int ticks)

Set the maximum number of minor tick intervals.

The scale's minor ticks are calculated automatically such that the number of minor intervals does not exceed ticks. The default value is 3.

Parameters

ticks | Maximal number of minor ticks.

See Also

scaleMaxMajor(), setScaleMaxMinor(), setScaleStepSize(), QwtScaleEngine::divideInterval()

12.4.3.24 void QwtAbstractScale::setScaleStepSize (double stepSize)

Set the step size used for calculating a scale division.

The step size is hint for calculating the intervals for the major ticks of the scale. A value of 0.0 is interpreted as no hint.

Parameters

stepSize Hint for the step size of the scale

See Also

scaleStepSize(), QwtScaleEngine::divideScale()

Note

Position and distance between the major ticks also depends on scaleMaxMajor().

12.4.3.25 void QwtAbstractScale::setUpperBound (double value)

Set the upper bound of the scale

Parameters

value Upper bound

See Also

upperBound(), setScale(), setLowerBound()

Note

For inverted scales the lower bound is greater than the upper bound

12.4.3.26 int QwtAbstractScale::transform (double value) const

Translate a scale value into a widget coordinate

Parameters

| value | Scale value |
|-------|-------------|
|-------|-------------|

Returns

Corresponding widget coordinate for value

See Also

scaleMap(), invTransform()

12.4.3.27 double QwtAbstractScale::upperBound () const

Returns

Upper bound of the scale

See Also

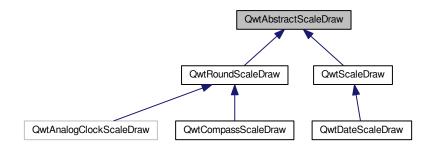
setUpperBound(), setScale(), lowerBound()

12.5 QwtAbstractScaleDraw Class Reference

A abstract base class for drawing scales.

#include <qwt_abstract_scale_draw.h>

Inheritance diagram for QwtAbstractScaleDraw:



Public Types

- enum ScaleComponent { Backbone = 0x01, Ticks = 0x02, Labels = 0x04 }
- typedef QFlags < ScaleComponent > ScaleComponents
 Scale components.

Public Member Functions

• QwtAbstractScaleDraw ()

Constructor.

virtual ~QwtAbstractScaleDraw ()

Destructor.

- void setScaleDiv (const QwtScaleDiv &s)
- · const QwtScaleDiv & scaleDiv () const
- void setTransformation (QwtTransform *)
- · const QwtScaleMap & scaleMap () const
- QwtScaleMap & scaleMap ()
- void enableComponent (ScaleComponent, bool enable=true)
- bool hasComponent (ScaleComponent) const
- void setTickLength (QwtScaleDiv::TickType, double length)
- double tickLength (QwtScaleDiv::TickType) const
- double maxTickLength () const
- void setSpacing (double margin)

Set the spacing between tick and labels.

• double spacing () const

Get the spacing.

void setPenWidth (int width)

Specify the width of the scale pen.

- int penWidth () const
- virtual void draw (QPainter *, const QPalette &) const

Draw the scale.

virtual QwtText label (double) const

Convert a value into its representing label.

- virtual double extent (const QFont &font) const =0
- void setMinimumExtent (double)

Set a minimum for the extent.

• double minimumExtent () const

Protected Member Functions

- virtual void drawTick (QPainter *painter, double value, double len) const =0
- virtual void drawBackbone (QPainter *painter) const =0
- virtual void drawLabel (QPainter *painter, double value) const =0
- void invalidateCache ()
- const QwtText & tickLabel (const QFont &, double value) const

Convert a value into its representing label and cache it.

12.5.1 Detailed Description

A abstract base class for drawing scales.

QwtAbstractScaleDraw can be used to draw linear or logarithmic scales.

After a scale division has been specified as a QwtScaleDiv object using setScaleDiv(), the scale can be drawn with the draw() member.

12.5.2 Member Enumeration Documentation

12.5.2.1 enum QwtAbstractScaleDraw::ScaleComponent

Components of a scale

See Also

enableComponent(), hasComponent

Enumerator

Backbone Backbone = the line where the ticks are located.

Ticks Ticks.

Labels Labels.

12.5.3 Constructor & Destructor Documentation

12.5.3.1 QwtAbstractScaleDraw::QwtAbstractScaleDraw()

Constructor.

The range of the scale is initialized to [0, 100], The spacing (distance between ticks and labels) is set to 4, the tick lengths are set to 4,6 and 8 pixels

12.5.4 Member Function Documentation

12.5.4.1 void QwtAbstractScaleDraw::draw (QPainter * painter, const QPalette & palette) const [virtual]

Draw the scale.

Parameters

| painter | The painter |
|---------|---|
| palette | Palette, text color is used for the labels, foreground color for ticks and backbone |

12.5.4.2 virtual void QwtAbstractScaleDraw::drawBackbone (QPainter * painter) const [protected], [pure virtual]

Draws the baseline of the scale

Parameters

| painter | Painter |
|---------|---------|
| | |

See Also

drawTick(), drawLabel()

Implemented in QwtScaleDraw, and QwtRoundScaleDraw.

12.5.4.3 virtual void QwtAbstractScaleDraw::drawLabel (QPainter * painter, double value) const [protected], [pure virtual]

Draws the label for a major scale tick

Parameters

| painter | Painter |
|---------|---------|
| value | Value |

See Also

drawTick(), drawBackbone()

Implemented in QwtScaleDraw, and QwtRoundScaleDraw.

12.5.4.4 virtual void QwtAbstractScaleDraw::drawTick (QPainter * painter, double value, double len) const [protected], [pure virtual]

Draw a tick

Parameters

| painter | Painter |
|---------|--------------------|
| value | Value of the tick |
| len | Length of the tick |

See Also

drawBackbone(), drawLabel()

Implemented in QwtScaleDraw, and QwtRoundScaleDraw.

12.5.4.5 void QwtAbstractScaleDraw::enableComponent (ScaleComponent component, bool enable = true)

En/Disable a component of the scale

Parameters

| component | Scale component |
|-----------|-----------------|
| enable | On/Off |

See Also

hasComponent()

12.5.4.6 virtual double QwtAbstractScaleDraw::extent (const QFont & font) const [pure virtual]

Calculate the extent

The extent is the distance from the baseline to the outermost pixel of the scale draw in opposite to its orientation. It is at least minimumExtent() pixels.

Parameters

| font | Font used for drawing the tick labels |
|------|---------------------------------------|

Returns

Number of pixels

See Also

setMinimumExtent(), minimumExtent()

Implemented in QwtScaleDraw, and QwtRoundScaleDraw.

12.5.4.7 bool QwtAbstractScaleDraw::hasComponent (ScaleComponent component) const

Check if a component is enabled

Parameters

| component | Component type |
|-----------|----------------|
| | 1 21 |

Returns

true, when component is enabled

See Also

enableComponent()

```
12.5.4.8 void QwtAbstractScaleDraw::invalidateCache() [protected]
```

Invalidate the cache used by tickLabel()

The cache is invalidated, when a new QwtScaleDiv is set. If the labels need to be changed. while the same QwtScaleDiv is set, invalidateCache() needs to be called manually.

12.5.4.9 QwtText QwtAbstractScaleDraw::label(double value) const [virtual]

Convert a value into its representing label.

The value is converted to a plain text using QLocale().toString(value). This method is often overloaded by applications to have individual labels.

Parameters

```
value Value
```

Returns

Label string.

Reimplemented in QwtDateScaleDraw, and QwtCompassScaleDraw.

12.5.4.10 double QwtAbstractScaleDraw::maxTickLength () const

Returns

Length of the longest tick

Useful for layout calculations

See Also

tickLength(), setTickLength()

12.5.4.11 double QwtAbstractScaleDraw::minimumExtent () const

Get the minimum extent

Returns

Minimum extent

See Also

extent(), setMinimumExtent()

12.5.4.12 int QwtAbstractScaleDraw::penWidth () const

Returns

Scale pen width

See Also

setPenWidth()

12.5.4.13 const QwtScaleDiv & QwtAbstractScaleDraw::scaleDiv () const

Returns

scale division

12.5.4.14 const QwtScaleMap & QwtAbstractScaleDraw::scaleMap () const

Returns

Map how to translate between scale and pixel values

12.5.4.15 QwtScaleMap & QwtAbstractScaleDraw::scaleMap ()

Returns

Map how to translate between scale and pixel values

12.5.4.16 void QwtAbstractScaleDraw::setMinimumExtent (double minExtent)

Set a minimum for the extent.

The extent is calculated from the components of the scale draw. In situations, where the labels are changing and the layout depends on the extent (f.e scrolling a scale), setting an upper limit as minimum extent will avoid jumps of the layout.

Parameters

| minExtent | Minimum extent |
|-----------|----------------|
|-----------|----------------|

See Also

extent(), minimumExtent()

12.5.4.17 void QwtAbstractScaleDraw::setPenWidth (int width)

Specify the width of the scale pen.

Parameters

| width | Pen width |
|-------|-----------|
|-------|-----------|

See Also

penWidth()

12.5.4.18 void QwtAbstractScaleDraw::setScaleDiv (const QwtScaleDiv & scaleDiv)

Change the scale division

Parameters

| scaleDiv | New scale division |
|----------|--------------------|

12.5.4.19 void QwtAbstractScaleDraw::setSpacing (double spacing)

Set the spacing between tick and labels.

The spacing is the distance between ticks and labels. The default spacing is 4 pixels.

Parameters

| spacing | Spacing |
|---------|---------|

See Also

spacing()

12.5.4.20 void QwtAbstractScaleDraw::setTickLength (QwtScaleDiv::TickType tickType, double length)

Set the length of the ticks

Parameters

| tickType | Tick type |
|----------|------------|
| length | New length |

Warning

the length is limited to [0..1000]

12.5.4.21 void QwtAbstractScaleDraw::setTransformation (QwtTransform * transformation)

Change the transformation of the scale

Parameters

| transformation | New scale transformation |
|----------------|--------------------------|
|----------------|--------------------------|

12.5.4.22 double QwtAbstractScaleDraw::spacing () const

Get the spacing.

The spacing is the distance between ticks and labels. The default spacing is 4 pixels.

Returns

Spacing

See Also

setSpacing()

12.5.4.23 const QwtText & QwtAbstractScaleDraw::tickLabel (const QFont & font, double value) const [protected]

Convert a value into its representing label and cache it.

The conversion between value and label is called very often in the layout and painting code. Unfortunately the calculation of the label sizes might be slow (really slow for rich text in Qt4), so it's necessary to cache the labels.

Parameters

| font | Font |
|-------|-------|
| value | Value |

Returns

Tick label

12.5.4.24 double QwtAbstractScaleDraw::tickLength (QwtScaleDiv::TickType tickType) const

Returns

Length of the ticks

See Also

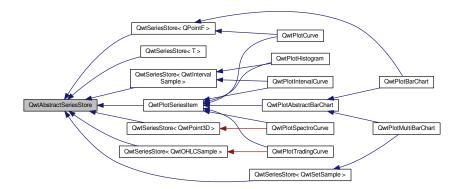
setTickLength(), maxTickLength()

12.6 QwtAbstractSeriesStore Class Reference

Bridge between QwtSeriesStore and QwtPlotSeriesItem.

#include <qwt_series_store.h>

Inheritance diagram for QwtAbstractSeriesStore:



Protected Member Functions

- virtual ~QwtAbstractSeriesStore ()
 - Destructor.
- virtual void dataChanged ()=0

dataChanged() indicates, that the series has been changed.

- virtual void setRectOfInterest (const QRectF &)=0
- virtual QRectF dataRect () const =0
- virtual size_t dataSize () const =0

12.6.1 Detailed Description

Bridge between QwtSeriesStore and QwtPlotSeriesItem.

QwtAbstractSeriesStore is an abstract interface only to make it possible to isolate the template based methods (QwtSeriesStore) from the regular methods (QwtPlotSeriesItem) to make it possible to derive from QwtPlotSeriesItem without any hassle with templates.

12.6.2 Member Function Documentation

12.6.2.1 virtual QRectF QwtAbstractSeriesStore::dataRect() const [protected], [pure virtual]

Returns

Bounding rectangle of the stored series

12.6.2.2 virtual size_t QwtAbstractSeriesStore::dataSize() const [protected], [pure virtual]

Returns

Number of samples

12.6.2.3 virtual void QwtAbstractSeriesStore::setRectOfInterest (const QRectF &) [protected], [pure virtual]

Set a the "rectangle of interest" for the stored series

See Also

QwtSeriesData<T>::setRectOfInterest()

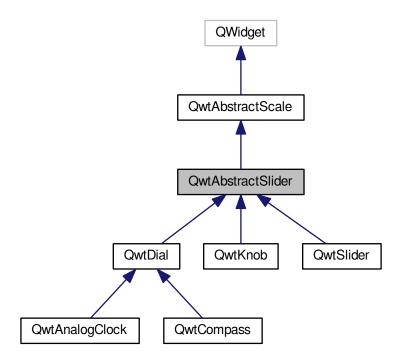
 $\label{local-loc$

12.7 QwtAbstractSlider Class Reference

An abstract base class for slider widgets with a scale.

#include <qwt_abstract_slider.h>

Inheritance diagram for QwtAbstractSlider:



Public Slots

• void setValue (double val)

Signals

void valueChanged (double value)

Notify a change of value.

- void sliderPressed ()
- void sliderReleased ()
- · void sliderMoved (double value)

Public Member Functions

QwtAbstractSlider (QWidget *parent=NULL)

Constructor.

virtual ~QwtAbstractSlider ()

Destructor.

- void setValid (bool)
- · bool isValid () const
- double value () const

Returns the current value.

- void setWrapping (bool)
- · bool wrapping () const
- void setTotalSteps (uint)

Set the number of steps.

- uint totalSteps () const
- void setSingleSteps (uint)

Set the number of steps for a single increment.

- uint singleSteps () const
- · void setPageSteps (uint)

Set the number of steps for a page increment.

- uint pageSteps () const
- void setStepAlignment (bool)

Enable step alignment.

- bool stepAlignment () const
- void setTracking (bool)

Enables or disables tracking.

- bool isTracking () const
- void setReadOnly (bool)
- bool isReadOnly () const
- void setInvertedControls (bool)
- bool invertedControls () const

Protected Member Functions

- virtual void mousePressEvent (QMouseEvent *)
- virtual void mouseReleaseEvent (QMouseEvent *)
- virtual void mouseMoveEvent (QMouseEvent *)
- virtual void keyPressEvent (QKeyEvent *)
- virtual void wheelEvent (QWheelEvent *)
- virtual bool isScrollPosition (const QPoint &pos) const =0

Determine what to do when the user presses a mouse button.

• virtual double scrolledTo (const QPoint &pos) const =0

Determine the value for a new position of the movable part of the slider.

- void incrementValue (int numSteps)
- virtual void scaleChange ()
- virtual void sliderChange ()

Calling update()

· double incrementedValue (double value, int stepCount) const

12.7.1 Detailed Description

An abstract base class for slider widgets with a scale.

A slider widget displays a value according to a scale. The class is designed as a common super class for widgets like QwtKnob, QwtDial and QwtSlider.

When the slider is nor readOnly() its value can be modified by keyboard, mouse and wheel inputs.

The range of the slider is divided into a number of steps from which the value increments according to user inputs depend. Only for linear scales the number of steps correspond with a fixed step size.

12.7.2 Constructor & Destructor Documentation

```
12.7.2.1 QwtAbstractSlider::QwtAbstractSlider(QWidget*parent=NULL) [explicit]
```

Constructor.

The scale is initialized to [0.0, 100.0], the number of steps is set to 100 with 1 and 10 and single an page step sizes. Step alignment is enabled.

The initial value is invalid.

Parameters

| parent | Parent widget |
|--------|---------------|

12.7.3 Member Function Documentation

12.7.3.1 double QwtAbstractSlider::incrementedValue (double value, int stepCount) const [protected]

Increment a value

Parameters

| val | ie | Value |
|---------|----|-----------------|
| stepCou | nt | Number of steps |

Returns

Incremented value

12.7.3.2 void QwtAbstractSlider::incrementValue (int stepCount) [protected]

Increment the slider

The step size depends on the number of totalSteps()

Parameters

| stepCount | Number of steps |
|-----------|-----------------|

See Also

setTotalSteps(), incrementedValue()

12.7.3.3 bool QwtAbstractSlider::invertedControls () const

Returns

True, when the controls are inverted

```
See Also
```

```
setInvertedControls()
```

12.7.3.4 bool QwtAbstractSlider::isReadOnly () const

In read only mode the slider can't be controlled by mouse or keyboard.

Returns

true if read only

See Also

setReadOnly()

12.7.3.5 virtual bool QwtAbstractSlider::isScrollPosition (const QPoint & pos) const [protected], [pure virtual]

Determine what to do when the user presses a mouse button.

Parameters

| pos | Mouse position |
|-----|----------------|
|-----|----------------|

Return values

| True, when | pos is a valid scroll position |
|------------|--------------------------------|
|------------|--------------------------------|

See Also

scrolledTo()

Implemented in QwtKnob, QwtDial, and QwtSlider.

12.7.3.6 bool QwtAbstractSlider::isTracking () const

Returns

True, when tracking has been enabled

See Also

setTracking()

12.7.3.7 bool QwtAbstractSlider::isValid () const

Returns

True, when the value is invalid

12.7.3.8 void QwtAbstractSlider::keyPressEvent (QKeyEvent * event) [protected], [virtual]

Handles key events

QwtAbstractSlider handles the following keys:

Qt::Key_Left

Add/Subtract singleSteps() in direction to lowerBound();

· Qt::Key_Right

Add/Subtract singleSteps() in direction to upperBound();

· Qt::Key_Down

Subtract singleSteps(), when invertedControls() is false

• Qt::Key_Up

Add singleSteps(), when invertedControls() is false

· Qt::Key_PageDown

Subtract pageSteps(), when invertedControls() is false

Qt::Key_PageUp

Add pageSteps(), when invertedControls() is false

· Qt::Key Home

Set the value to the minimum()

· Qt::Key_End

Set the value to the maximum()

Parameters

| avant | Koy ovent |
|-------|------------|
| event | reveren |
| 0.0 | 1.05 0.0.0 |

See Also

isReadOnly()

Reimplemented in QwtCompass.

12.7.3.9 void QwtAbstractSlider::mouseMoveEvent (QMouseEvent * event) [protected], [virtual]

Mouse Move Event handler

Parameters

event Mouse event

12.7.3.10 void QwtAbstractSlider::mousePressEvent (QMouseEvent * **event**) [protected], [virtual]

Mouse press event handler

Parameters

event Mouse event

Reimplemented in QwtSlider.

12.7.3.11 void QwtAbstractSlider::mouseReleaseEvent (QMouseEvent * event) [protected], [virtual]

Mouse Release Event handler

Parameters

event Mouse event

Reimplemented in QwtSlider.

12.7.3.12 uint QwtAbstractSlider::pageSteps () const

Returns

Number of steps

See Also

```
setPageSteps(), totalSteps(), singleSteps()
```

12.7.3.13 void QwtAbstractSlider::scaleChange() [protected], [virtual]

Update the slider according to modifications of the scale

Reimplemented from QwtAbstractScale.

Reimplemented in QwtDial, and QwtSlider.

12.7.3.14 virtual double QwtAbstractSlider::scrolledTo (const QPoint & pos) const [protected], [pure virtual]

Determine the value for a new position of the movable part of the slider.

Parameters

pos Mouse position

Returns

Value for the mouse position

See Also

isScrollPosition()

Implemented in QwtKnob, QwtDial, and QwtSlider.

12.7.3.15 void QwtAbstractSlider::setInvertedControls (bool on)

Invert wheel and key events

Usually scrolling the mouse wheel "up" and using keys like page up will increase the slider's value towards its maximum. When invertedControls() is enabled the value is scrolled towards its minimum.

Inverting the controls might be f.e. useful for a vertical slider with an inverted scale (decreasing from top to bottom).

Parameters

on Invert controls, when true

See Also

invertedControls(), keyEvent(), wheelEvent()

12.7.3.16 void QwtAbstractSlider::setPageSteps (uint stepCount)

Set the number of steps for a page increment.

The range of the slider is divided into a number of steps from which the value increments according to user inputs depend.

Parameters

stepCount Number of steps

See Also

pageSteps(), setTotalSteps(), setSingleSteps()

12.7.3.17 void QwtAbstractSlider::setReadOnly (bool on)

En/Disable read only mode

In read only mode the slider can't be controlled by mouse or keyboard.

Parameters

on Enables in case of true

See Also

isReadOnly()

Warning

The focus policy is set to Qt::StrongFocus or Qt::NoFocus

12.7.3.18 void QwtAbstractSlider::setSingleSteps (uint stepCount)

Set the number of steps for a single increment.

The range of the slider is divided into a number of steps from which the value increments according to user inputs depend.

Parameters

stepCount Number of steps

See Also

singleSteps(), setTotalSteps(), setPageSteps()

12.7.3.19 void QwtAbstractSlider::setStepAlignment (bool on)

Enable step alignment.

When step alignment is enabled values resulting from slider movements are aligned to the step size.

Parameters

on Enable step alignment when true

See Also

stepAlignment()

12.7.3.20 void QwtAbstractSlider::setTotalSteps (uint stepCount)

Set the number of steps.

The range of the slider is divided into a number of steps from which the value increments according to user inputs depend.

The default setting is 100.

Parameters

stepCount Number of steps

See Also

totalSteps(), setSingleSteps(), setPageSteps()

12.7.3.21 void QwtAbstractSlider::setTracking (bool on)

Enables or disables tracking.

If tracking is enabled, the slider emits the valueChanged() signal while the movable part of the slider is being dragged. If tracking is disabled, the slider emits the valueChanged() signal only when the user releases the slider.

Tracking is enabled by default.

Parameters

on true (enable) or false (disable) tracking.

See Also

isTracking(), sliderMoved()

12.7.3.22 void QwtAbstractSlider::setValid (bool on)

Set the value to be valid/invalid

Parameters

on When true, the value is invalidated

See Also

setValue()

12.7.3.23 void QwtAbstractSlider::setValue (double value) [slot]

Set the slider to the specified value

Parameters

value New value

See Also

setValid(), sliderChange(), valueChanged()

12.7.3.24 void QwtAbstractSlider::setWrapping (bool on)

If wrapping is true stepping up from upperBound() value will take you to the minimum() value and vice versa.

Parameters

on En/Disable wrapping

See Also

wrapping()

```
uint QwtAbstractSlider::singleSteps ( ) const
Returns
      Number of steps
See Also
      setSingleSteps(), totalSteps(), pageSteps()
12.7.3.26 void QwtAbstractSlider::sliderMoved ( double value ) [signal]
This signal is emitted when the user moves the slider with the mouse.
Parameters
                     New value
             value
See Also
      valueChanged()
12.7.3.27 void QwtAbstractSlider::sliderPressed() [signal]
This signal is emitted when the user presses the movable part of the slider.
12.7.3.28 void QwtAbstractSlider::sliderReleased() [signal]
This signal is emitted when the user releases the movable part of the slider.
12.7.3.29 bool QwtAbstractSlider::stepAlignment ( ) const
Returns
      True, when step alignment is enabled
See Also
      setStepAlignment()
12.7.3.30 uint QwtAbstractSlider::totalSteps ( ) const
Returns
      Number of steps
See Also
      setTotalSteps(), singleSteps(), pageSteps()
12.7.3.31 void QwtAbstractSlider::valueChanged ( double value ) [signal]
Notify a change of value.
When tracking is enabled (default setting), this signal will be emitted every time the value changes.
```

Parameters

| value | New value |
|-------|-----------|
|-------|-----------|

See Also

setTracking(), sliderMoved()

12.7.3.32 void QwtAbstractSlider::wheelEvent (QWheelEvent * event) [protected], [virtual]

Wheel Event handler

In/decreases the value by s number of steps. The direction depends on the invertedControls() property.

When the control or shift modifier is pressed the wheel delta (divided by 120) is mapped to an increment according to pageSteps(). Otherwise it is mapped to singleSteps().

Parameters

Reimplemented in QwtDial.

12.7.3.33 bool QwtAbstractSlider::wrapping () const

Returns

True, when wrapping is set

See Also

setWrapping()

12.8 QwtAlphaColorMap Class Reference

QwtAlphaColorMap varies the alpha value of a color.

#include <qwt_color_map.h>

Inheritance diagram for QwtAlphaColorMap:



Public Member Functions

- QwtAlphaColorMap (const QColor &=QColor(Qt::gray))
- virtual ~QwtAlphaColorMap ()

Destructor.

- void setColor (const QColor &)
- · QColor color () const
- virtual QRgb rgb (const QwtInterval &, double value) const

Map a value of a given interval into a alpha value.

Additional Inherited Members

12.8.1 Detailed Description

QwtAlphaColorMap varies the alpha value of a color.

12.8.2 Constructor & Destructor Documentation

12.8.2.1 QwtAlphaColorMap::QwtAlphaColorMap (const QColor & color = QColor (Qt::gray))

Constructor

Parameters

| color Color of the map |
|--------------------------|
|--------------------------|

12.8.3 Member Function Documentation

12.8.3.1 QColor QwtAlphaColorMap::color () const

Returns

the color

See Also

setColor()

12.8.3.2 QRgb QwtAlphaColorMap::rgb (const QwtInterval & interval, double value) const [virtual]

Map a value of a given interval into a alpha value.

alpha := (value - interval.minValue()) / interval.width();

Parameters

| interval | Range for all values |
|----------|-------------------------------|
| value | Value to map into a RGB value |

Returns

RGB value, with an alpha value

Implements QwtColorMap.

12.8.3.3 void QwtAlphaColorMap::setColor (const QColor & color)

Set the color

Parameters

| color | Color |
|-------|-------|
|-------|-------|

See Also

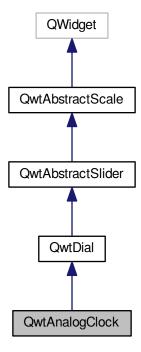
color()

12.9 QwtAnalogClock Class Reference

An analog clock.

#include <qwt_analog_clock.h>

Inheritance diagram for QwtAnalogClock:



Public Types

• enum Hand { SecondHand, MinuteHand, HourHand, NHands }

Public Slots

void setCurrentTime ()

Set the current time.

void setTime (const QTime &)

Public Member Functions

- QwtAnalogClock (QWidget *parent=NULL)
- virtual ~QwtAnalogClock ()

Destructor.

- void setHand (Hand, QwtDialNeedle *)
- const QwtDialNeedle * hand (Hand) const
- QwtDialNeedle * hand (Hand)

Protected Member Functions

virtual void drawNeedle (QPainter *, const QPointF &, double radius, double direction, QPalette::ColorGroup)

Draw the needle.

 virtual void drawHand (QPainter *, Hand, const QPointF &, double radius, double direction, QPalette::Color-Group) const

Additional Inherited Members

12.9.1 Detailed Description

An analog clock.

Example

```
#include <qwt_analog_clock.h>

QwtAnalogClock *clock = new QwtAnalogClock(...);
clock->scaleDraw()->setPenWidth(3);
clock->setLineWidth(6);
clock->setFrameShadow(QwtDial::Sunken);
clock->setTime();

// update the clock every second
QTimer *timer = new QTimer(clock);
timer->connect(timer, SIGNAL(timeout()), clock, SLOT(setCurrentTime()));
timer->start(1000);
```

Note

The examples/dials example shows how to use QwtAnalogClock.

12.9.2 Member Enumeration Documentation

12.9.2.1 enum QwtAnalogClock::Hand

Hand type

See Also

```
setHand(), hand()
```

Enumerator

SecondHand Needle displaying the seconds.

MinuteHand Needle displaying the minutes.

HourHand Needle displaying the hours.

NHands Number of needles.

12.9.3 Constructor & Destructor Documentation

12.9.3.1 QwtAnalogClock::QwtAnalogClock (QWidget * parent = NULL) [explicit]

Constructor

Parameters

| parent | Parent widget |
|--------|---------------|
|--------|---------------|

12.9.4 Member Function Documentation

12.9.4.1 void QwtAnalogClock::drawHand (QPainter * painter, Hand hd, const QPointF & center, double radius, double direction, QPalette::ColorGroup cg) const [protected], [virtual]

Draw a clock hand

Parameters

| painter | Painter |
|-----------|---|
| hd | Specify the type of hand |
| center | Center of the clock |
| radius | Maximum length for the hands |
| direction | Direction of the hand in degrees, counter clockwise |
| cg | ColorGroup |

12.9.4.2 void QwtAnalogClock::drawNeedle (QPainter * painter, const QPointF & center, double radius, double dir, QPalette::ColorGroup colorGroup) const [protected], [virtual]

Draw the needle.

A clock has no single needle but three hands instead. drawNeedle() translates value() into directions for the hands and calls drawHand().

Parameters

| painter | Painter |
|------------|------------------------------|
| center | Center of the clock |
| radius | Maximum length for the hands |
| dir | Dummy, not used. |
| colorGroup | ColorGroup |

See Also

drawHand()

Reimplemented from QwtDial.

12.9.4.3 const QwtDialNeedle * QwtAnalogClock::hand (Hand hd) const

Returns

Clock hand

Parameters

| hd | Specifies the type of hand |
|----|----------------------------|

See Also

setHand()

12.9.4.4 QwtDialNeedle * QwtAnalogClock::hand (Hand hd)

Returns

Clock hand

Parameters

| hd | Specifies the type of hand |
|----|----------------------------|
|----|----------------------------|

See Also

setHand()

12.9.4.5 void QwtAnalogClock::setHand (Hand hand, QwtDialNeedle * needle)

Set a clock hand

Parameters

| hand | Specifies the type of hand |
|--------|----------------------------|
| needle | Hand |

See Also

hand()

12.9.4.6 void QwtAnalogClock::setTime (const QTime & time) [slot]

Set a time

Parameters

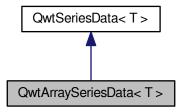
| time | Time to display |
|------|-----------------|

12.10 QwtArraySeriesData < T > Class Template Reference

Template class for data, that is organized as QVector.

#include <qwt_series_data.h>

Inheritance diagram for QwtArraySeriesData< T >:



Public Member Functions

• QwtArraySeriesData ()

Constructor.

- QwtArraySeriesData (const QVector< T > &samples)
- void setSamples (const QVector< T > &samples)

```
    const QVector< T > samples () const
```

- virtual size_t size () const
- virtual T sample (size_t index) const

Protected Attributes

QVector < T > d_samples
 Vector of samples.

12.10.1 Detailed Description

template<typename T>class QwtArraySeriesData<T>

Template class for data, that is organized as QVector.

QVector uses implicit data sharing and can be passed around as argument efficiently.

12.10.2 Constructor & Destructor Documentation

12.10.2.1 template < typename T > QwtArraySeriesData < T > :: QwtArraySeriesData (const QVector < T > & samples)

Constructor

Parameters

samples Array of samples

12.10.3 Member Function Documentation

12.10.3.1 template < typename T > T QwtArraySeriesData < T >::sample (size_t index) const [virtual]

Returns

Sample at a specific position

Parameters

| index | Index |
|-------|-------|

Returns

Sample at position index

Implements QwtSeriesData< T >.

12.10.3.2 template < typename T > const QVector < T > QwtArraySeriesData < T >::samples () const

Returns

Array of samples

12.10.3.3 template < typename T > void QwtArraySeriesData < T >::setSamples (const QVector < T > & samples)

Assign an array of samples

Parameters

samples Array of samples

12.10.3.4 template < typename T > size_t QwtArraySeriesData < T >::size() const [virtual]

Returns

Number of samples

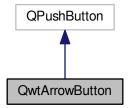
Implements QwtSeriesData< T >.

12.11 QwtArrowButton Class Reference

Arrow Button.

#include <qwt_arrow_button.h>

Inheritance diagram for QwtArrowButton:



Public Member Functions

- QwtArrowButton (int num, Qt::ArrowType, QWidget *parent=NULL)
- virtual ~QwtArrowButton ()

Destructor.

Qt::ArrowType arrowType () const

The direction of the arrows.

• int num () const

The number of arrows.

- virtual QSize sizeHint () const
- · virtual QSize minimumSizeHint () const

Return a minimum size hint.

Protected Member Functions

- virtual void paintEvent (QPaintEvent *event)
- virtual void drawButtonLabel (QPainter *p)

Draw the button label.

- virtual void drawArrow (QPainter *, const QRect &, Qt::ArrowType) const
- virtual QRect labelRect () const
- virtual QSize arrowSize (Qt::ArrowType, const QSize &boundingSize) const

virtual void keyPressEvent (QKeyEvent *)
 autoRepeat for the space keys

12.11.1 Detailed Description

Arrow Button.

A push button with one or more filled triangles on its front. An Arrow button can have 1 to 3 arrows in a row, pointing up, down, left or right.

12.11.2 Constructor & Destructor Documentation

Parameters

| num | Number of arrows |
|-----------|-----------------------------------|
| arrowType | see Qt::ArrowType in the Qt docs. |
| parent | Parent widget |

12.11.3 Member Function Documentation

12.11.3.1 QSize QwtArrowButton::arrowSize (Qt::ArrowType arrowType, const QSize & boundingSize) const [protected], [virtual]

Calculate the size for a arrow that fits into a rectangle of a given size

Parameters

| arrowType | Arrow type |
|--------------|---------------|
| boundingSize | Bounding size |

Returns

Size of the arrow

12.11.3.2 void QwtArrowButton::drawArrow (QPainter * painter, const QRect & r, Qt::ArrowType arrowType) const [protected], [virtual]

Draw an arrow int a bounding rectangle

Parameters

| painter | Painter |
|-----------|------------------------------------|
| r | Rectangle where to paint the arrow |
| arrowType | Arrow type |

12.11.3.3 void QwtArrowButton::drawButtonLabel(QPainter * painter) [protected], [virtual]

Draw the button label.

Parameters

Generated on Thu Sep 18 2014 16:59:51 for Qwt User's Guide by Doxygen

painter Painter

See Also

The Qt Manual for QPushButton

12.11.3.4 QRect QwtArrowButton::labelRect() const [protected], [virtual]

Returns

the bounding rectangle for the label

12.11.3.5 void QwtArrowButton::paintEvent (QPaintEvent * event) [protected], [virtual]

Paint event handler

Parameters

event Paint event

12.11.3.6 QSize QwtArrowButton::sizeHint() const [virtual]

Returns

a size hint

12.12 QwtClipper Class Reference

Some clipping algorithms.

```
#include <qwt_clipper.h>
```

Static Public Member Functions

- static QPolygon clipPolygon (const QRect &, const QPolygon &, bool closePolygon=false)
- static QPolygon clipPolygon (const QRectF &, const QPolygon &, bool closePolygon=false)
- static QPolygonF clipPolygonF (const QRectF &, const QPolygonF &, bool closePolygon=false)
- static QVector< QwtInterval > clipCircle (const QRectF &, const QPointF &, double radius)

12.12.1 Detailed Description

Some clipping algorithms.

12.12.2 Member Function Documentation

12.12.2.1 QVector < QwtInterval > QwtClipper::clipCircle (const QRectF & clipRect, const QPointF & center, double radius) [static]

Circle clipping

clipCircle() divides a circle into intervals of angles representing arcs of the circle. When the circle is completely inside the clip rectangle an interval $[0.0, 2 * M_PI]$ is returned.

Parameters

| clipRect | Clip rectangle |
|----------|----------------------|
| center | Center of the circle |
| radius | Radius of the circle |

Returns

Arcs of the circle

12.12.2.2 QPolygon QwtClipper::clipPolygon (const QRect & clipRect, const QPolygon & polygon, bool closePolygon = false) [static]

Sutherland-Hodgman polygon clipping

Parameters

| clipRect | Clip rectangle |
|--------------|----------------------------------|
| polygon | Polygon |
| closePolygon | True, when the polygon is closed |

Returns

Clipped polygon

12.12.2.3 QPolygon QwtClipper::clipPolygon (const QRectF & clipRect, const QPolygon & polygon, bool closePolygon = false) [static]

Sutherland-Hodgman polygon clipping

Parameters

| clipRect | Clip rectangle |
|--------------|----------------------------------|
| polygon | Polygon |
| closePolygon | True, when the polygon is closed |

Returns

Clipped polygon

12.12.2.4 QPolygonF QwtClipper::clipPolygonF (const QRectF & clipRect, const QPolygonF & polygon, bool closePolygon = false) [static]

Sutherland-Hodgman polygon clipping

Parameters

| clipRect | Clip rectangle |
|--------------|----------------------------------|
| polygon | Polygon |
| closePolygon | True, when the polygon is closed |

Returns

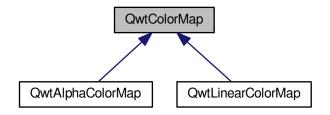
Clipped polygon

12.13 QwtColorMap Class Reference

QwtColorMap is used to map values into colors.

#include <qwt_color_map.h>

Inheritance diagram for QwtColorMap:



Public Types

• enum Format { RGB, Indexed }

Public Member Functions

• QwtColorMap (Format=QwtColorMap::RGB)

Constructor.

virtual ~QwtColorMap ()

Destructor.

- · Format format () const
- virtual QRgb rgb (const QwtInterval &interval, double value) const =0
- virtual unsigned char colorIndex (const QwtInterval &interval, double value) const =0
- QColor color (const QwtInterval &, double value) const
- virtual QVector< QRgb > colorTable (const QwtInterval &) const

12.13.1 Detailed Description

QwtColorMap is used to map values into colors.

For displaying 3D data on a 2D plane the 3rd dimension is often displayed using colors, like f.e in a spectrogram.

Each color map is optimized to return colors for only one of the following image formats:

- QImage::Format_Indexed8
- · QImage::Format_ARGB32

See Also

QwtPlotSpectrogram, QwtScaleWidget

12.13.2 Member Enumeration Documentation

12.13.2.1 enum QwtColorMap::Format

Format for color mapping

See Also

rgb(), colorIndex(), colorTable()

Enumerator

RGB The map is intended to map into RGB values.

Indexed The map is intended to map into 8 bit values, that are indices into the color table.

12.13.3 Member Function Documentation

12.13.3.1 QColor QwtColorMap::color (const QwtInterval & interval, double value) const [inline]

Map a value into a color

Parameters

| interval | Valid interval for values |
|----------|---------------------------|
| value | Value |

Returns

Color corresponding to value

Warning

This method is slow for Indexed color maps. If it is necessary to map many values, its better to get the color table once and find the color using colorIndex().

12.13.3.2 virtual unsigned char QwtColorMap::colorIndex (const QwtInterval & interval, double value) const [pure virtual]

Map a value of a given interval into a color index

Parameters

| interval | Range for the values |
|----------|----------------------|
| value | Value |

Returns

color index, corresponding to value

Implemented in QwtLinearColorMap.

12.13.3.3 QVector < QRgb > QwtColorMap::colorTable (const QwtInterval & interval) const [virtual]

Build and return a color map of 256 colors

The color table is needed for rendering indexed images in combination with using colorIndex().

Parameters

| interval | Range for the values |
|----------|----------------------|

Returns

A color table, that can be used for a QImage

12.13.3.4 QwtColorMap::Format QwtColorMap::format() const [inline]

Returns

Intended format of the color map

See Also

Format

12.13.3.5 virtual QRgb QwtColorMap::rgb (const QwtInterval & interval, double value) const [pure virtual]

Map a value of a given interval into a RGB value.

Parameters

| interval | Range for the values |
|----------|----------------------|
| value | Value |

Returns

RGB value, corresponding to value

Implemented in QwtAlphaColorMap, and QwtLinearColorMap.

12.14 QwtColumnRect Class Reference

Directed rectangle representing bounding rectangle and orientation of a column.

```
#include <qwt_column_symbol.h>
```

Public Types

enum Direction { LeftToRight, RightToLeft, BottomToTop, TopToBottom }
 Direction of the column.

Public Member Functions

• QwtColumnRect ()

Build an rectangle with invalid intervals directed BottomToTop.

- QRectF toRect () const
- · Qt::Orientation orientation () const

Public Attributes

QwtInterval hInterval

Interval for the horizontal coordinates.

· QwtInterval vInterval

Interval for the vertical coordinates.

· Direction direction

Direction.

12.14.1 Detailed Description

Directed rectangle representing bounding rectangle and orientation of a column.

12.14.2 Member Enumeration Documentation

12.14.2.1 enum QwtColumnRect::Direction

Direction of the column.

Enumerator

LeftToRight From left to right.RightToLeft From right to left.BottomToTop From bottom to top.TopToBottom From top to bottom.

12.14.3 Member Function Documentation

12.14.3.1 Qt::Orientation QwtColumnRect::orientation () const [inline]

Returns

Orientation

12.14.3.2 QRectF QwtColumnRect::toRect() const [inline]

Returns

A normalized QRect built from the intervals

12.15 QwtColumnSymbol Class Reference

A drawing primitive for columns.

```
#include <qwt_column_symbol.h>
```

Public Types

- enum Style { NoStyle = -1, Box, UserStyle = 1000 }
- enum FrameStyle { NoFrame, Plain, Raised }

Public Member Functions

- QwtColumnSymbol (Style=NoStyle)
- virtual \sim QwtColumnSymbol ()

Destructor.

- void setFrameStyle (FrameStyle style)
- FrameStyle frameStyle () const
- void setLineWidth (int width)
- int lineWidth () const
- void setPalette (const QPalette &)
- const QPalette & palette () const
- void setStyle (Style)
- Style style () const
- virtual void draw (QPainter *, const QwtColumnRect &) const

Protected Member Functions

void drawBox (QPainter *, const QwtColumnRect &) const

```
12.15.1 Detailed Description
A drawing primitive for columns.
 12.15.2 Member Enumeration Documentation
12.15.2.1 enum QwtColumnSymbol::FrameStyle
Frame Style used in Box style().
 See Also
      Style, setFrameStyle(), frameStyle(), setStyle(), setPalette()
Enumerator
     NoFrame No frame.
     Plain A plain frame style.
     Raised A raised frame style.
12.15.2.2 enum QwtColumnSymbol::Style
Style
 See Also
      setStyle(), style()
Enumerator
     NoStyle No Style, the symbol draws nothing.
     Box The column is painted with a frame depending on the frameStyle() and lineWidth() using the palette().
     UserStyle Styles >= QwtColumnSymbol::UserStyle are reserved for derived classes of QwtColumnSymbol
          that overload draw() with additional application specific symbol types.
12.15.3 Constructor & Destructor Documentation
12.15.3.1 QwtColumnSymbol::QwtColumnSymbol ( Style style = NoStyle )
 Constructor
 Parameters
               style | Style of the symbol
 See Also
```

```
setStyle(), style(), Style
```

12.15.4 Member Function Documentation

12.15.4.1 void QwtColumnSymbol::draw (QPainter * painter, const QwtColumnRect & rect) const [virtual]

Draw the symbol depending on its style.

Parameters

| painter | Painter |
|---------|--------------------|
| rect | Directed rectangle |

See Also

drawBox()

12.15.4.2 void QwtColumnSymbol::drawBox (QPainter * painter, const QwtColumnRect & rect) const [protected]

Draw the symbol when it is in Box style.

Parameters

| painter | Painter |
|---------|--------------------|
| rect | Directed rectangle |

See Also

draw()

12.15.4.3 QwtColumnSymbol::FrameStyle QwtColumnSymbol::frameStyle () const

Returns

Current frame style, that is used for the Box style.

See Also

setFrameStyle(), lineWidth(), setStyle()

12.15.4.4 int QwtColumnSymbol::lineWidth () const

Returns

Line width of the frame, that is used for the Box style.

See Also

setLineWidth(), frameStyle(), setStyle()

12.15.4.5 const QPalette & QwtColumnSymbol::palette () const

Returns

Current palette

See Also

setPalette()

12.15.4.6 void QwtColumnSymbol::setFrameStyle (FrameStyle frameStyle)

Set the frame, that is used for the Box style.

Parameters

frameStyle Frame style

See Also

frameStyle(), setLineWidth(), setStyle()

12.15.4.7 void QwtColumnSymbol::setLineWidth (int width)

Set the line width of the frame, that is used for the Box style.

Parameters

width Width

See Also

lineWidth(), setFrameStyle()

12.15.4.8 void QwtColumnSymbol::setPalette (const QPalette & palette)

Assign a palette for the symbol

Parameters

palette Palette

See Also

palette(), setStyle()

12.15.4.9 void QwtColumnSymbol::setStyle (Style style)

Specify the symbol style

Parameters

style Style

See Also

style(), setPalette()

 $12.15.4.10 \quad \textbf{QwtColumnSymbol::Style QwtColumnSymbol::style (\ \) const$

Returns

Current symbol style

See Also

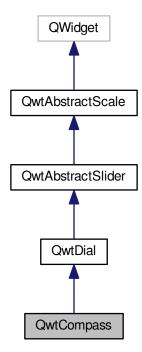
setStyle()

12.16 QwtCompass Class Reference

A Compass Widget.

#include <qwt_compass.h>

Inheritance diagram for QwtCompass:



Public Member Functions

QwtCompass (QWidget *parent=NULL)

Constructor.

virtual ~QwtCompass ()

Destructor.

- void setRose (QwtCompassRose *rose)
- const QwtCompassRose * rose () const
- QwtCompassRose * rose ()

Protected Member Functions

- virtual void drawRose (QPainter *, const QPointF ¢er, double radius, double north, QPalette::ColorGroup)
 const
- virtual void drawScaleContents (QPainter *, const QPointF ¢er, double radius) const
- virtual void keyPressEvent (QKeyEvent *)

Additional Inherited Members

12.16.1 Detailed Description

A Compass Widget.

QwtCompass is a widget to display and enter directions. It consists of a scale, an optional needle and rose.

Note

The examples/dials example shows how to use QwtCompass.

12.16.2 Constructor & Destructor Documentation

12.16.2.1 QwtCompass::QwtCompass (QWidget * parent = NULL) [explicit]

Constructor.

Parameters

| parent | Parent widget |
|--------|---------------|

Create a compass widget with a scale, no needle and no rose. The default origin is 270.0 with no valid value. It accepts mouse and keyboard inputs and has no step size. The default mode is QwtDial::RotateNeedle.

12.16.3 Member Function Documentation

12.16.3.1 void QwtCompass::drawRose (QPainter * painter, const QPointF & center, double radius, double north, QPalette::ColorGroup cg) const [protected], [virtual]

Draw the compass rose

Parameters

| painter | Painter |
|---------|--|
| center | Center of the compass |
| radius | of the circle, where to paint the rose |
| north | Direction pointing north, in degrees counter clockwise |
| cg | Color group |

12.16.3.2 void QwtCompass::drawScaleContents (QPainter * painter, const QPointF & center, double radius) const [protected], [virtual]

Draw the contents of the scale

Parameters

| painter | Painter |
|---------|------------------------------|
| center | Center of the content circle |
| radius | Radius of the content circle |

Reimplemented from QwtDial.

12.16.3.3 void QwtCompass::keyPressEvent (QKeyEvent * kev) [protected], [virtual]

Handles key events

Beside the keys described in QwtDial::keyPressEvent numbers from 1-9 (without 5) set the direction according to their position on the num pad.

See Also

isReadOnly()

Reimplemented from QwtAbstractSlider.

12.16.3.4 const QwtCompassRose * QwtCompass::rose () const

Returns

rose

```
See Also
```

setRose()

12.16.3.5 QwtCompassRose * QwtCompass::rose ()

Returns

rose

See Also

setRose()

12.16.3.6 void QwtCompass::setRose (QwtCompassRose * rose)

Set a rose for the compass

Parameters

rose Compass rose

Warning

The rose will be deleted, when a different rose is set or in \sim QwtCompass

See Also

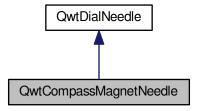
rose()

12.17 QwtCompassMagnetNeedle Class Reference

A magnet needle for compass widgets.

```
#include <qwt_dial_needle.h>
```

Inheritance diagram for QwtCompassMagnetNeedle:



Public Types

• enum Style { TriangleStyle, ThinStyle }

Style of the needle.

Public Member Functions

QwtCompassMagnetNeedle (Style=TriangleStyle, const QColor &light=Qt::white, const QColor &dark=Qt::red)

Constructor.

Protected Member Functions

• virtual void drawNeedle (QPainter *, double length, QPalette::ColorGroup) const

12.17.1 Detailed Description

A magnet needle for compass widgets.

A magnet needle points to two opposite directions indicating north and south.

The following colors are used:

QPalette::Light
 Used for pointing south

QPalette::Dark
 Used for pointing north

QPalette::Base
 Knob (ThinStyle only)

See Also

QwtDial, QwtCompass

12.17.2 Member Enumeration Documentation

12.17.2.1 enum QwtCompassMagnetNeedle::Style

Style of the needle.

Enumerator

TriangleStyle A needle with a triangular shape.

ThinStyle A thin needle.

12.17.3 Member Function Documentation

12.17.3.1 void QwtCompassMagnetNeedle::drawNeedle (QPainter * painter, double length, QPalette::ColorGroup colorGroup) const [protected], [virtual]

Draw the needle

Parameters

| painter |
|---------|
|---------|

| length | Length of the needle |
|------------|--------------------------------|
| colorGroup | Color group, used for painting |

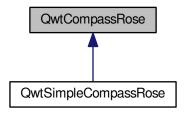
Implements QwtDialNeedle.

12.18 QwtCompassRose Class Reference

Abstract base class for a compass rose.

#include <qwt_compass_rose.h>

Inheritance diagram for QwtCompassRose:



Public Member Functions

virtual ~QwtCompassRose ()

Destructor.

virtual void setPalette (const QPalette &p)

Assign a palette.

- const QPalette & palette () const
- virtual void draw (QPainter *painter, const QPointF ¢er, double radius, double north, QPalette::Color-Group colorGroup=QPalette::Active) const =0

12.18.1 Detailed Description

Abstract base class for a compass rose.

12.18.2 Member Function Documentation

12.18.2.1 virtual void QwtCompassRose::draw (QPainter * painter, const QPointF & center, double radius, double north,

QPalette::ColorGroup = QPalette::Active) const [pure virtual]

Draw the rose

Parameters

| painter | Painter |
|---------|---------|

| center | Center point |
|------------|--------------------|
| radius | Radius of the rose |
| north | Position |
| colorGroup | Color group |

Implemented in QwtSimpleCompassRose.

12.18.2.2 const QPalette& QwtCompassRose::palette() const [inline]

Returns

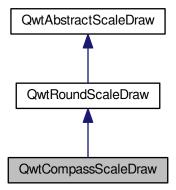
Current palette

12.19 QwtCompassScaleDraw Class Reference

A special scale draw made for QwtCompass.

#include <qwt_compass.h>

Inheritance diagram for QwtCompassScaleDraw:



Public Member Functions

• QwtCompassScaleDraw ()

Constructor.

QwtCompassScaleDraw (const QMap< double, QString > &map)

Constructor.

void setLabelMap (const QMap< double, QString > &map)

Set a map, mapping values to labels.

- QMap< double, QString > labelMap () const
- · virtual QwtText label (double value) const

Additional Inherited Members

12.19.1 Detailed Description

A special scale draw made for QwtCompass.

QwtCompassScaleDraw maps values to strings using a special map, that can be modified by the application The default map consists of the labels N, NE, E, SE, S, SW, W, NW.

See Also

QwtCompass

12.19.2 Constructor & Destructor Documentation

12.19.2.1 QwtCompassScaleDraw::QwtCompassScaleDraw() [explicit]

Constructor.

Initializes a label map for multiples of 45 degrees

12.19.2.2 QwtCompassScaleDraw::QwtCompassScaleDraw (const QMap < double, QString > & map) [explicit]

Constructor.

Parameters

map Value to label map

12.19.3 Member Function Documentation

12.19.3.1 QwtText QwtCompassScaleDraw::label (double value) const [virtual]

Map a value to a corresponding label

Parameters

value Value that will be mapped

label() looks in the labelMap() for a corresponding label for value or returns an null text.

Returns

Label, or QString::null

See Also

labelMap(), setLabelMap()

Reimplemented from QwtAbstractScaleDraw.

12.19.3.2 QMap < double, QString > QwtCompassScaleDraw::labelMap () const

Returns

map, mapping values to labels

See Also

setLabelMap()

12.19.3.3 void QwtCompassScaleDraw::setLabelMap (const QMap < double, QString > & map)

Set a map, mapping values to labels.

Parameters

| тар | Value to label map |
|-----|--------------------|
|-----|--------------------|

The values of the major ticks are found by looking into this map. The default map consists of the labels N, NE, E, SE, S, SW, W, NW.

Warning

The map will have no effect for values that are no major tick values. Major ticks can be changed by QwtScale-Draw::setScale

See Also

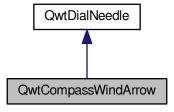
labelMap(), scaleDraw(), setScale()

12.20 QwtCompassWindArrow Class Reference

An indicator for the wind direction.

#include <qwt_dial_needle.h>

Inheritance diagram for QwtCompassWindArrow:



Public Types

• enum Style { Style1, Style2 } Style of the arrow.

Public Member Functions

• QwtCompassWindArrow (Style, const QColor &light=Qt::white, const QColor &dark=Qt::gray)

Protected Member Functions

• virtual void drawNeedle (QPainter *, double length, QPalette::ColorGroup) const

12.20.1 Detailed Description

An indicator for the wind direction.

QwtCompassWindArrow shows the direction where the wind comes from.

· QPalette::Light

Used for Style1, or the light half of Style2

· QPalette::Dark

Used for the dark half of Style2

See Also

QwtDial, QwtCompass

12.20.2 Member Enumeration Documentation

12.20.2.1 enum QwtCompassWindArrow::Style

Style of the arrow.

Enumerator

Style1 A needle pointing to the center.

Style2 A needle pointing to the center.

12.20.3 Constructor & Destructor Documentation

12.20.3.1 QwtCompassWindArrow:QwtCompassWindArrow (Style style, const QColor & light = Qt::white, const QColor & dark = Qt::gray)

Constructor

Parameters

| style | Arrow style |
|-------|-------------|
| light | Light color |
| dark | Dark color |

12.20.4 Member Function Documentation

12.20.4.1 void QwtCompassWindArrow::drawNeedle (QPainter * painter, double length, QPalette::ColorGroup colorGroup) const [protected], [virtual]

Draw the needle

Parameters

| painter | Painter |
|------------|--------------------------------|
| length | Length of the needle |
| colorGroup | Color group, used for painting |

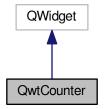
Implements QwtDialNeedle.

12.21 QwtCounter Class Reference

The Counter Widget.

#include <qwt_counter.h>

Inheritance diagram for QwtCounter:



Public Types

enum Button { Button1, Button2, Button3, ButtonCnt }
 Button index.

Public Slots

• void setValue (double)

Set a new value without adjusting to the step raster.

Signals

- void buttonReleased (double value)
- void valueChanged (double value)

Public Member Functions

- QwtCounter (QWidget *parent=NULL)
- virtual ~QwtCounter ()

Destructor.

- void setValid (bool)
- · bool isValid () const
- void setWrapping (bool)

En/Disable wrapping.

- bool wrapping () const
- bool isReadOnly () const
- void setReadOnly (bool)

Allow/disallow the user to manually edit the value.

- void setNumButtons (int n)
- int numButtons () const
- void setIncSteps (QwtCounter::Button btn, int nSteps)
- int incSteps (QwtCounter::Button btn) const
- virtual QSize sizeHint () const

A size hint.

- double singleStep () const
- void setSingleStep (double s)

Set the step size of the counter.

void setRange (double min, double max)

Set the minimum and maximum values.

- double minimum () const
- void setMinimum (double min)
- · double maximum () const
- void setMaximum (double max)
- void setStepButton1 (int nSteps)
- int stepButton1 () const

returns the number of increment steps for button 1

- void setStepButton2 (int nSteps)
- int stepButton2 () const

returns the number of increment steps for button 2

- void setStepButton3 (int nSteps)
- int stepButton3 () const

returns the number of increment steps for button 3

· double value () const

Protected Member Functions

- virtual bool event (QEvent *)
- virtual void wheelEvent (QWheelEvent *)
- virtual void keyPressEvent (QKeyEvent *)

12.21.1 Detailed Description

The Counter Widget.

A Counter consists of a label displaying a number and one ore more (up to three) push buttons on each side of the label which can be used to increment or decrement the counter's value.

A counter has a range from a minimum value to a maximum value and a step size. When the wrapping property is set the counter is circular.

The number of steps by which a button increments or decrements the value can be specified using setIncSteps(). The number of buttons can be changed with setNumButtons().

Example:

12.21.2 Member Enumeration Documentation

12.21.2.1 enum QwtCounter::Button

Button index.

Enumerator

Button1 Button intended for minor steps.

Button2 Button intended for medium steps.

Button3 Button intended for large steps.

ButtonCnt Number of buttons.

12.21.3 Constructor & Destructor Documentation

```
12.21.3.1 QwtCounter::QwtCounter(QWidget*parent=NULL) [explicit]
```

The counter is initialized with a range is set to [0.0, 1.0] with 0.01 as single step size. The value is invalid.

The default number of buttons is set to 2. The default increments are:

• Button 1: 1 step

· Button 2: 10 steps

• Button 3: 100 steps

Parameters

parent

12.21.4 Member Function Documentation

12.21.4.1 void QwtCounter::buttonReleased (double value) [signal]

This signal is emitted when a button has been released

Parameters

value The new value

12.21.4.2 bool QwtCounter::event (QEvent * event) [protected], [virtual]

Handle QEvent::PolishRequest events

Parameters

event Event

Returns

see QWidget::event()

12.21.4.3 int QwtCounter::incSteps (QwtCounter::Button button) const

Returns

The number of steps by which a specified button increments the value or 0 if the button is invalid.

Parameters

button | Button index

See Also

setIncSteps()

```
12.21.4.4 bool QwtCounter::isReadOnly ( ) const
Returns
     True, when the line line edit is read only. (default is no)
See Also
     setReadOnly()
12.21.4.5 bool QwtCounter::isValid ( ) const
Returns
     True, if the value is valid
See Also
     setValid(), setValue()
12.21.4.6 void QwtCounter::keyPressEvent ( QKeyEvent * event ) [protected], [virtual]
Handle key events
    · Ctrl + Qt::Key_Home
      Step to minimum()
    · Ctrl + Qt::Key_End
      Step to maximum()

    Qt::Key_Up

      Increment by incSteps(QwtCounter::Button1)
    · Qt::Key_Down
      Decrement by incSteps(QwtCounter::Button1)
    · Qt::Key_PageUp
      Increment by incSteps(QwtCounter::Button2)
    · Qt::Key_PageDown
      Decrement by incSteps(QwtCounter::Button2)

    Shift + Qt::Key_PageUp

      Increment by incSteps(QwtCounter::Button3)
    · Shift + Qt::Key PageDown
      Decrement by incSteps(QwtCounter::Button3)
Parameters
             event
                     Key event
12.21.4.7 double QwtCounter::maximum ( ) const
Returns
     The maximum of the range
See Also
```

setRange(), setMaximum(), minimum()

12.21.4.8 double QwtCounter::minimum () const

Returns

The minimum of the range

See Also

setRange(), setMinimum(), maximum()

12.21.4.9 int QwtCounter::numButtons () const

Returns

The number of buttons on each side of the widget.

See Also

setNumButtons()

12.21.4.10 void QwtCounter::setIncSteps (QwtCounter::Button button, int numSteps)

Specify the number of steps by which the value is incremented or decremented when a specified button is pushed.

Parameters

| button | Button index |
|----------|-----------------|
| numSteps | Number of steps |

See Also

incSteps()

12.21.4.11 void QwtCounter::setMaximum (double value)

Set the maximum value of the range

Parameters

| value Maximum value | |
|-----------------------|--|
|-----------------------|--|

See Also

setRange(), setMinimum(), maximum()

12.21.4.12 void QwtCounter::setMinimum (double value)

Set the minimum value of the range

Parameters

| value | Minimum value |
|-------|---------------|

See Also

setRange(), setMaximum(), minimum()

Note

The maximum is adjusted if necessary to ensure that the range remains valid.

12.21.4.13 void QwtCounter::setNumButtons (int numButtons)

Specify the number of buttons on each side of the label

Parameters

| numButtons | Number of buttons |
|------------|-------------------|
|------------|-------------------|

See Also

numButtons()

12.21.4.14 void QwtCounter::setRange (double min, double max)

Set the minimum and maximum values.

The maximum is adjusted if necessary to ensure that the range remains valid. The value might be modified to be inside of the range.

Parameters

| min | Minimum value |
|-----|---------------|
| max | Maximum value |

See Also

minimum(), maximum()

12.21.4.15 void QwtCounter::setReadOnly (bool on)

Allow/disallow the user to manually edit the value.

Parameters

| on | True disable editing |
|----|----------------------|
|----|----------------------|

See Also

isReadOnly()

12.21.4.16 void QwtCounter::setSingleStep (double stepSize)

Set the step size of the counter.

A value <= 0.0 disables stepping

Parameters

| stepSize Single step size |
|-----------------------------|
|-----------------------------|

See Also

singleStep()

12.21.4.17 void QwtCounter::setStepButton1 (int nSteps)

Set the number of increment steps for button 1

Parameters

| nSteps Number of steps |
|------------------------|
|------------------------|

12.21.4.18 void QwtCounter::setStepButton2 (int nSteps)

Set the number of increment steps for button 2

Parameters

nSteps Number of steps

12.21.4.19 void QwtCounter::setStepButton3 (int nSteps)

Set the number of increment steps for button 3

Parameters

nSteps Number of steps

12.21.4.20 void QwtCounter::setValid (bool on)

Set the counter to be in valid/invalid state

When the counter is set to invalid, no numbers are displayed and the buttons are disabled.

Parameters

on If true the counter will be set as valid

See Also

setValue(), isValid()

12.21.4.21 void QwtCounter::setValue (double value) [slot]

Set a new value without adjusting to the step raster.

The state of the counter is set to be valid.

Parameters

value New value

See Also

isValid(), value(), valueChanged()

Warning

The value is clipped when it lies outside the range.

12.21.4.22 void QwtCounter::setWrapping (bool on)

En/Disable wrapping.

If wrapping is true stepping up from maximum() value will take you to the minimum() value and vice versa.

Parameters

on En/Disable wrapping

See Also

wrapping()

12.21.4.23 double QwtCounter::singleStep () const

Returns

Single step size

See Also

setSingleStep()

12.21.4.24 double QwtCounter::value () const

Returns

Current value of the counter

See Also

setValue(), valueChanged()

12.21.4.25 void QwtCounter::valueChanged (double value) [signal]

This signal is emitted when the counter's value has changed

Parameters

value | The new value

12.21.4.26 void QwtCounter::wheelEvent (QWheelEvent * event) [protected], [virtual]

Handle wheel events

Parameters

event Wheel event

12.21.4.27 bool QwtCounter::wrapping () const

Returns

True, when wrapping is set

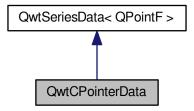
See Also

setWrapping()

12.22 QwtCPointerData Class Reference

Data class containing two pointers to memory blocks of doubles.

Inheritance diagram for QwtCPointerData:



Public Member Functions

- QwtCPointerData (const double *x, const double *y, size_t size)
- virtual QRectF boundingRect () const

Calculate the bounding rectangle.

- virtual size_t size () const
- virtual QPointF sample (size t i) const
- const double * xData () const
- const double * yData () const

Additional Inherited Members

12.22.1 Detailed Description

Data class containing two pointers to memory blocks of doubles.

12.22.2 Constructor & Destructor Documentation

12.22.2.1 QwtCPointerData::QwtCPointerData (const double * x, const double * y, size_t size)

Constructor

Parameters

| X | Array of x values |
|---|-------------------|
| У | Array of y values |

| size | Size of the x and y arrays |
|------|----------------------------|
| | |

Warning

The programmer must assure that the memory blocks referenced by the pointers remain valid during the lifetime of the QwtPlotCPointer object.

See Also

```
QwtPlotCurve::setData(), QwtPlotCurve::setRawSamples()
```

12.22.3 Member Function Documentation

```
12.22.3.1 QRectF QwtCPointerData::boundingRect() const [virtual]
```

Calculate the bounding rectangle.

The bounding rectangle is calculated once by iterating over all points and is stored for all following requests.

Returns

Bounding rectangle

```
Implements QwtSeriesData < QPointF >.
```

12.22.3.2 QPointF QwtCPointerData::sample (size_t index) const [virtual]

Return the sample at position i

Parameters

```
index Index
```

Returns

Sample at position i

```
Implements\ QwtSeriesData < QPointF>.
```

12.22.3.3 size_t QwtCPointerData::size() const [virtual]

Returns

Size of the data set

Implements QwtSeriesData < QPointF >.

12.22.3.4 const double * QwtCPointerData::xData () const

Returns

Array of the x-values

12.22.3.5 const double * QwtCPointerData::yData () const

Returns

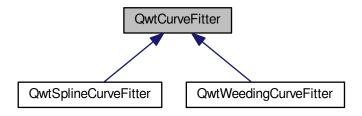
Array of the y-values

12.23 QwtCurveFitter Class Reference

Abstract base class for a curve fitter.

#include <qwt_curve_fitter.h>

Inheritance diagram for QwtCurveFitter:



Public Member Functions

virtual ~QwtCurveFitter ()

Destructor.

virtual QPolygonF fitCurve (const QPolygonF &polygon) const =0

Protected Member Functions

• QwtCurveFitter ()

Constructor.

12.23.1 Detailed Description

Abstract base class for a curve fitter.

12.23.2 Member Function Documentation

12.23.2.1 virtual QPolygonF QwtCurveFitter::fitCurve (const QPolygonF & polygon) const [pure virtual]

Find a curve which has the best fit to a series of data points

Parameters

| polygon | Series of data points |
|---------|-----------------------|

Returns

Curve points

Implemented in QwtWeedingCurveFitter, and QwtSplineCurveFitter.

12.24 QwtDate Class Reference

A collection of methods around date/time values.

```
#include <qwt_date.h>
```

Public Types

- enum Week0Type { FirstThursday, FirstDay }
- enum IntervalType {
 Millisecond, Second, Minute, Hour,
 Day, Week, Month, Year }
- enum { JulianDayForEpoch = 2440588 }

Static Public Member Functions

- static QDate minDate ()
- static QDate maxDate ()
- static QDateTime toDateTime (double value, Qt::TimeSpec=Qt::UTC)
- static double to Double (const QDateTime &)
- static QDateTime ceil (const QDateTime &, IntervalType)
- static QDateTime floor (const QDateTime &, IntervalType)
- static QDate dateOfWeek0 (int year, Week0Type)

Date of the first day of the first week for a year.

- static int weekNumber (const QDate &, Week0Type)
- static int utcOffset (const QDateTime &)
- static QString toString (const QDateTime &, const QString &format, Week0Type)

12.24.1 Detailed Description

A collection of methods around date/time values.

Qt offers convenient classes for dealing with date/time values, but Qwt uses coordinate systems that are based on doubles. QwtDate offers methods to translate from QDateTime to double and v.v.

A double is interpreted as the number of milliseconds since 1970-01-01T00:00:00 Universal Coordinated Time - also known as "The Epoch".

While the range of the Julian day in Qt4 is limited to [0, MAX_INT], Qt5 stores it as qint64 offering a huge range of valid dates. As the significance of a double is below this (assuming a fraction of 52 bits) the translation is not bijective with rounding errors for dates very far from Epoch. For a resolution of 1 ms those start to happen for dates above the year 144683.

An axis for a date/time interval is expected to be aligned and divided in time/date units like seconds, minutes, ... QwtDate offers several algorithms that are needed to calculate these axes.

See Also

QwtDateScaleEngine, QwtDateScaleDraw, QDate, QTime

12.24.2 Member Enumeration Documentation

12.24.2.1 anonymous enum

Enumerator

JulianDayForEpoch The Julian day of "The Epoch".

12.24.2.2 enum QwtDate::IntervalType

Classification of an time interval

Time intervals needs to be classified to decide how to align and divide it.

Enumerator

Millisecond The interval is related to milliseconds.

Second The interval is related to seconds.

Minute The interval is related to minutes.

Hour The interval is related to hours.

Day The interval is related to days.

Week The interval is related to weeks.

Month The interval is related to months.

Year The interval is related to years.

12.24.2.3 enum QwtDate::Week0Type

How to identify the first week of year differs between countries.

Enumerator

FirstThursday According to ISO 8601 the first week of a year is defined as "the week with the year's first Thursday in it".

FirstThursday corresponds to the numbering that is implemented in QDate::weekNumber().

FirstDay "The week with January 1.1 in it."

In the U.S. this definition is more common than FirstThursday.

12.24.3 Member Function Documentation

12.24.3.1 QDateTime QwtDate::ceil (const QDateTime & dateTime, IntervalType intervalType) [static]

Ceil a datetime according the interval type

Parameters

| dateTime | Datetime value |
|--------------|---|
| intervalType | Interval type, how to ceil. F.e. when intervalType = QwtDate::Months, the result will be ceiled |
| | to the next beginning of a month |

Returns

Ceiled datetime

See Also

floor()

12.24.3.2 QDate QwtDate::dateOfWeek0 (int year, Week0Type type) [static]

Date of the first day of the first week for a year.

The first day of a week depends on the current locale (QLocale::firstDayOfWeek()).

Parameters

| year | Year |
|------|---------------------------------------|
| type | Option how to identify the first week |

Returns

First day of week 0

See Also

QLocale::firstDayOfWeek(), weekNumber()

12.24.3.3 QDateTime QwtDate::floor (const QDateTime & dateTime, IntervalType intervalType) [static]

Floor a datetime according the interval type

Parameters

| dateTime | Datetime value |
|--------------|---|
| intervalType | Interval type, how to ceil. F.e. when intervalType = QwtDate::Months, the result will be ceiled |
| | to the next beginning of a month |

Returns

Floored datetime

See Also

floor()

12.24.3.4 QDate QwtDate::maxDate() [static]

Maximum for the supported date range

The range of valid dates depends on how QDate stores the Julian day internally.

- For Qt4 it is "Tue Jun 3 5874898"
- For Qt5 it is "Tue Dec 31 2147483647"

Returns

maximum of the date range

See Also

minDate()

Note

The maximum differs between Qt4 and Qt5

12.24.3.5 QDate QwtDate::minDate() [static]

Minimum for the supported date range

The range of valid dates depends on how QDate stores the Julian day internally.

- For Qt4 it is "Tue Jan 2 -4713"
- For Qt5 it is "Thu Jan 1 -2147483648"

Returns

minimum of the date range

See Also

maxDate()

12.24.3.6 QDateTime QwtDate::toDateTime (double value, Qt::TimeSpec timeSpec = Qt::UTC) [static]

Translate from double to QDateTime

Parameters

| value | Number of milliseconds since the epoch, 1970-01-01T00:00:00 UTC |
|----------|---|
| timeSpec | Time specification |

Returns

Datetime value

See Also

toDouble(), QDateTime::setMSecsSinceEpoch()

Note

The return datetime for Qt::OffsetFromUTC will be Qt::UTC

12.24.3.7 double QwtDate::toDouble (const QDateTime & dateTime) [static]

Translate from QDateTime to double

Parameters

| dateTime | Datetime value |
|----------|----------------|
|----------|----------------|

Returns

Number of milliseconds since 1970-01-01T00:00:00 UTC has passed.

See Also

toDateTime(), QDateTime::toMSecsSinceEpoch()

Warning

For values very far below or above 1970-01-01 UTC rounding errors will happen due to the limited significance of a double.

12.24.3.8 QString QwtDate::toString (const QDateTime & dateTime, const QString & format, Week0Type week0Type) [static]

Translate a datetime into a string

Beside the format expressions documented in QDateTime::toString() the following expressions are supported:

w
 week number: (1 - 53)

• ww

week number with a leading zero (01 - 53)

Parameters

| dateTime | Datetime value |
|-----------|-------------------------|
| format | Format string |
| week0Type | Specification of week 0 |

Returns

Datetime string

See Also

QDateTime::toString(), weekNumber(), QwtDateScaleDraw

12.24.3.9 int QwtDate::utcOffset (const QDateTime & dateTime) [static]

Offset in seconds from Coordinated Universal Time

The offset depends on the time specification of dateTime:

- Qt::UTC 0, dateTime has no offset
- · Qt::OffsetFromUTC returns dateTime.utcOffset()
- · Qt::LocalTime: number of seconds from the UTC

For Qt::LocalTime the offset depends on the timezone and daylight savings.

Parameters

| dateTime |
|----------|
|----------|

Returns

Offset in seconds

12.24.3.10 int QwtDate::weekNumber (const QDate & date, WeekOType type) [static]

Find the week number of a date

QwtDate::FirstThursday
 Corresponding to ISO 8601 (see QDate::weekNumber()).

QwtDate::FirstDay

Number of weeks that have begun since dateOfWeek0().

Parameters

| date | Date |
|------|---------------------------------------|
| type | Option how to identify the first week |

Returns

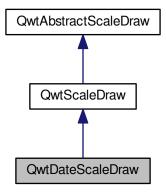
Week number, starting with 1

12.25 QwtDateScaleDraw Class Reference

A class for drawing datetime scales.

```
#include <qwt_date_scale_draw.h>
```

Inheritance diagram for QwtDateScaleDraw:



Public Member Functions

• QwtDateScaleDraw (Qt::TimeSpec=Qt::LocalTime)

Constructor.

virtual ~QwtDateScaleDraw ()

Destructor.

- void setDateFormat (QwtDate::IntervalType, const QString &)
- QString dateFormat (QwtDate::IntervalType) const
- void setTimeSpec (Qt::TimeSpec)
- Qt::TimeSpec timeSpec () const
- void setUtcOffset (int seconds)
- int utcOffset () const
- void setWeek0Type (QwtDate::Week0Type)
- QwtDate::Week0Type week0Type () const
- virtual QwtText label (double) const

Convert a value into its representing label.

QDateTime toDateTime (double) const

Protected Member Functions

- virtual QwtDate::IntervalType intervalType (const QwtScaleDiv &) const
- virtual QString dateFormatOfDate (const QDateTime &, QwtDate::IntervalType) const

Additional Inherited Members

12.25.1 Detailed Description

A class for drawing datetime scales.

QwtDateScaleDraw displays values as datetime labels. The format of the labels depends on the alignment of the major tick labels.

The default format strings are:

- · Millisecond
 - "hh:mm:ss:zzz\nddd dd MMM yyyy"
- Second
 - "hh:mm:ss\nddd dd MMM yyyy"
- Minute
 - "hh:mm\nddd dd MMM yyyy"
- Hour
 - "hh:mm\nddd dd MMM yyyy"
- Day
 - "ddd dd MMM yyyy"
- Week
 - "Www yyyy"
- Month
 - "MMM yyyy"
- Year
 - "уууу"

The format strings can be modified using setDateFormat() or individually for each tick label by overloading date-FormatOfDate(),

Usually QwtDateScaleDraw is used in combination with QwtDateScaleEngine, that calculates scales for datetime intervals.

See Also

QwtDateScaleEngine, QwtPlot::setAxisScaleDraw()

12.25.2 Constructor & Destructor Documentation

12.25.2.1 QwtDateScaleDraw::QwtDateScaleDraw (Qt::TimeSpec timeSpec = Qt :: LocalTime)

Constructor.

The default setting is to display tick labels for the given time specification. The first week of a year is defined like for QwtDate::FirstThursday.

Parameters

| timeSpec | Time specification |
|----------|--------------------|
|----------|--------------------|

See Also

setTimeSpec(), setWeek0Type()

12.25.3 Member Function Documentation

12.25.3.1 QString QwtDateScaleDraw::dateFormat (QwtDate::IntervalType intervalType) const

Parameters

| intervalType | Interval type |
|--------------|---------------|

Returns

Default format string for an datetime interval type

See Also

setDateFormat(), dateFormatOfDate()

12.25.3.2 QString QwtDateScaleDraw::dateFormatOfDate (const QDateTime & dateTime, QwtDate::IntervalType intervalType) const [protected], [virtual]

Format string for the representation of a datetime

dateFormatOfDate() is intended to be overloaded for situations, where formats are individual for specific datetime values.

The default setting ignores dateTime and return the default format for the interval type.

Parameters

| dateTime | Datetime value |
|--------------|----------------|
| intervalType | Interval type |

Returns

Format string

See Also

setDateFormat(), QwtDate::toString()

12.25.3.3 QwtDate::IntervalType QwtDateScaleDraw::intervalType (const QwtScaleDiv & scaleDiv) const [protected], [virtual]

Find the less detailed datetime unit, where no rounding errors happen.

Parameters

| scaleDiv Scale division |
|-------------------------|
|-------------------------|

Returns

Interval type

See Also

dateFormatOfDate()

12.25.3.4 QwtText QwtDateScaleDraw::label (double value) const [virtual]

Convert a value into its representing label.

The value is converted to a datetime value using to Date Time() and converted to a plain text using QwtDate::to-String().

Parameters

| value | Value |
|-------|-------|

Returns

Label string.

See Also

dateFormatOfDate()

Reimplemented from QwtAbstractScaleDraw.

12.25.3.5 void QwtDateScaleDraw::setDateFormat (QwtDate::IntervalType intervalType, const QString & format)

Set the default format string for an datetime interval type

Parameters

| intervalType | Interval type |
|--------------|-----------------------|
| format | Default format string |

See Also

dateFormat(), dateFormatOfDate(), QwtDate::toString()

12.25.3.6 void QwtDateScaleDraw::setTimeSpec (Qt::TimeSpec timeSpec)

Set the time specification used for the tick labels

Parameters

| timeS | Spec | Time specification |
|-------|------|--------------------|

See Also

timeSpec(), setUtcOffset(), toDateTime()

12.25.3.7 void QwtDateScaleDraw::setUtcOffset (int seconds)

Set the offset in seconds from Coordinated Universal Time

Parameters

| seconds Offset in seconds |
|---------------------------|
|---------------------------|

Note

The offset has no effect beside for the time specification Qt::OffsetFromUTC.

See Also

QDate::utcOffset(), setTimeSpec(), toDateTime()

12.25.3.8 void QwtDateScaleDraw::setWeek0Type (QwtDate::Week0Type week0Type)

Sets how to identify the first week of a year.

```
Parameters
```

week0Type | Mode how to identify the first week of a year

See Also

week0Type().

Note

week0Type has no effect beside for intervals classified as QwtDate::Week.

12.25.3.9 Qt::TimeSpec QwtDateScaleDraw::timeSpec () const

Returns

Time specification used for the tick labels

See Also

setTimeSpec(), utcOffset(), toDateTime()

12.25.3.10 QDateTime QwtDateScaleDraw::toDateTime (double value) const

Translate a double value into a QDateTime object.

Returns

QDateTime object initialized with timeSpec() and utcOffset().

See Also

timeSpec(), utcOffset(), QwtDate::toDateTime()

12.25.3.11 int QwtDateScaleDraw::utcOffset () const

Returns

Offset in seconds from Coordinated Universal Time

Note

The offset has no effect beside for the time specification Qt::OffsetFromUTC.

See Also

QDate::setUtcOffset(), setTimeSpec(), toDateTime()

12.25.3.12 QwtDate::Week0Type QwtDateScaleDraw::week0Type () const

Returns

Setting how to identify the first week of a year.

See Also

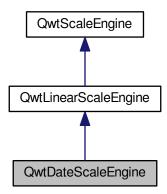
setWeek0Type()

12.26 QwtDateScaleEngine Class Reference

A scale engine for date/time values.

#include <qwt_date_scale_engine.h>

Inheritance diagram for QwtDateScaleEngine:



Public Member Functions

QwtDateScaleEngine (Qt::TimeSpec=Qt::LocalTime)

Constructor.

virtual ~QwtDateScaleEngine ()

Destructor.

- void setTimeSpec (Qt::TimeSpec)
- Qt::TimeSpec timeSpec () const
- void setUtcOffset (int seconds)
- int utcOffset () const
- void setWeek0Type (QwtDate::Week0Type)
- QwtDate::Week0Type week0Type () const
- void setMaxWeeks (int)
- int maxWeeks () const
- virtual void autoScale (int maxNumSteps, double &x1, double &x2, double &stepSize) const
- virtual QwtScaleDiv divideScale (double x1, double x2, int maxMajorSteps, int maxMinorSteps, double step-Size=0.0) const

Calculate a scale division for a date/time interval.

- virtual QwtDate::IntervalType intervalType (const QDateTime &, const QDateTime &, int maxSteps) const
- QDateTime toDateTime (double) const

Protected Member Functions

virtual QDateTime alignDate (const QDateTime &, double stepSize, QwtDate::IntervalType, bool up) const

Additional Inherited Members

12.26.1 Detailed Description

A scale engine for date/time values.

QwtDateScaleEngine builds scales from a time intervals. Together with QwtDateScaleDraw it can be used for axes according to date/time values.

Years, months, weeks, days, hours and minutes are organized in steps with non constant intervals. QwtDateScale-Engine classifies intervals and aligns the boundaries and tick positions according to this classification.

QwtDateScaleEngine supports representations depending on Qt::TimeSpec specifications. The valid range for scales is limited by the range of QDateTime, that differs between Qt4 and Qt5.

Datetime values are expected as the number of milliseconds since 1970-01-01T00:00:00 Universal Coordinated Time - also known as "The Epoch", that can be converted to QDateTime using QwtDate::toDateTime().

See Also

QwtDate, QwtPlot::setAxisScaleEngine(), QwtAbstractScale::setScaleEngine()

12.26.2 Constructor & Destructor Documentation

12.26.2.1 QwtDateScaleEngine::QwtDateScaleEngine (Qt::TimeSpec timeSpec = Qt::LocalTime)

Constructor.

The engine is initialized to build scales for the given time specification. It classifies intervals > 4 weeks as >= Qt::Month. The first week of a year is defined like for QwtDate::FirstThursday.

Parameters

| timeSpec | Time specification |
|----------|--------------------|
|----------|--------------------|

See Also

setTimeSpec(), setMaxWeeks(), setWeek0Type()

12.26.3 Member Function Documentation

12.26.3.1 QDateTime QwtDateScaleEngine::alignDate (const QDateTime & dateTime, double stepSize, QwtDate::IntervalType intervalType, bool up) const [protected], [virtual]

Align a date/time value for a step size

For Qt::Day alignments there is no "natural day 0" - instead the first day of the year is used to avoid jumping major ticks positions when panning a scale. For other alignments (f.e according to the first day of the month) alignDate() has to be overloaded.

Parameters

| dateTime | Date/time value |
|--------------|--|
| stepSize | Step size |
| intervalType | Interval type |
| ир | When true dateTime is ceiled - otherwise it is floored |

Returns

Aligned date/time value

12.26.3.2 void QwtDateScaleEngine::autoScale (int *maxNumSteps*, double & *x1*, double & *x2*, double & *stepSize*) const [virtual]

Align and divide an interval

The algorithm aligns and divides the interval into steps.

Datetime interval divisions are usually not equidistant and the calculated stepSize can only be used as an approximation for the steps calculated by divideScale().

Parameters

| maxNumSteps | Max. number of steps |
|-------------|---------------------------------------|
| x1 | First limit of the interval (In/Out) |
| x2 | Second limit of the interval (In/Out) |
| stepSize | Step size (Out) |

See Also

QwtScaleEngine::setAttribute()

Reimplemented from QwtLinearScaleEngine.

12.26.3.3 QwtScaleDiv QwtDateScaleEngine::divideScale (double x1, double x2, int maxMajorSteps, int maxMinorSteps, double stepSize = 0.0) const [virtual]

Calculate a scale division for a date/time interval.

Parameters

| x1 | First interval limit |
|---------------|--|
| x2 | Second interval limit |
| maxMajorSteps | Maximum for the number of major steps |
| maxMinorSteps | Maximum number of minor steps |
| stepSize | Step size. If stepSize == 0, the scaleEngine calculates one. |

Returns

Calculated scale division

Reimplemented from QwtLinearScaleEngine.

12.26.3.4 QwtDate::IntervalType QwtDateScaleEngine::intervalType (const QDateTime & minDate, const QDateTime & maxDate, int maxSteps) const [virtual]

Classification of a date/time interval division

Parameters

| minDate | Minimum (= earlier) of the interval |
|----------|---------------------------------------|
| maxDate | Maximum (= later) of the interval |
| maxSteps | Maximum for the number of steps |

Returns

Interval classification

12.26.3.5 int QwtDateScaleEngine::maxWeeks () const

Returns

Upper limit for the number of weeks, when an interval can be classified as Qt::Week.

See Also

setMaxWeeks(), week0Type()

12.26.3.6 void QwtDateScaleEngine::setMaxWeeks (int weeks)

Set a upper limit for the number of weeks, when an interval can be classified as Qt::Week.

The default setting is 4 weeks.

Parameters

weeks Upper limit for the number of weeks

Note

In business charts a year is often devided into weeks [1-52]

See Also

maxWeeks(), setWeek0Type()

12.26.3.7 void QwtDateScaleEngine::setTimeSpec (Qt::TimeSpec timeSpec)

Set the time specification used by the engine

Parameters

timeSpec Time specification

See Also

timeSpec(), setUtcOffset(), toDateTime()

12.26.3.8 void QwtDateScaleEngine::setUtcOffset (int seconds)

Set the offset in seconds from Coordinated Universal Time

Parameters

seconds Offset in seconds

Note

The offset has no effect beside for the time specification Qt::OffsetFromUTC.

See Also

QDate::utcOffset(), setTimeSpec(), toDateTime()

12.26.3.9 void QwtDateScaleEngine::setWeek0Type (QwtDate::Week0Type week0Type)

Sets how to identify the first week of a year.

Parameters

```
Mode how to identify the first week of a year
       week0Type
See Also
      week0Type(), setMaxWeeks()
Note
      week0Type has no effect beside for intervals classified as QwtDate::Week.
12.26.3.10 Qt::TimeSpec QwtDateScaleEngine::timeSpec ( ) const
Returns
      Time specification used by the engine
See Also
      setTimeSpec(), utcOffset(), toDateTime()
12.26.3.11 QDateTime QwtDateScaleEngine::toDateTime ( double value ) const
Translate a double value into a QDateTime object.
For QDateTime result is bounded by QwtDate::minDate() and QwtDate::maxDate()
Returns
      QDateTime object initialized with timeSpec() and utcOffset().
See Also
      timeSpec(), utcOffset(), QwtDate::toDateTime()
12.26.3.12 int QwtDateScaleEngine::utcOffset ( ) const
Returns
      Offset in seconds from Coordinated Universal Time
Note
      The offset has no effect beside for the time specification Qt::OffsetFromUTC.
See Also
      QDate::setUtcOffset(), setTimeSpec(), toDateTime()
12.26.3.13 QwtDate::Week0Type QwtDateScaleEngine::week0Type ( ) const
Returns
      Setting how to identify the first week of a year.
See Also
```

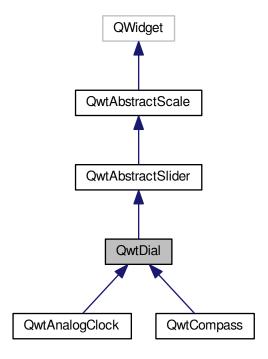
setWeek0Type(), maxWeeks()

12.27 QwtDial Class Reference

QwtDial class provides a rounded range control.

#include <qwt_dial.h>

Inheritance diagram for QwtDial:



Public Types

- enum Shadow { Plain = QFrame::Plain, Raised = QFrame::Raised, Sunken = QFrame::Sunken }
 Frame shadow.
- enum Mode { RotateNeedle, RotateScale }

Mode controlling whether the needle or the scale is rotating.

Public Member Functions

- QwtDial (QWidget *parent=NULL)
 - Constructor.
- virtual ~QwtDial ()

Destructor.

- · void setFrameShadow (Shadow)
- Shadow frameShadow () const
- void setLineWidth (int)
- int lineWidth () const
- void setMode (Mode)

Change the mode of the dial.

- Mode mode () const
- void setScaleArc (double min, double max)
- void setMinScaleArc (double min)
- · double minScaleArc () const
- void setMaxScaleArc (double min)
- · double maxScaleArc () const
- · virtual void setOrigin (double)

Change the origin.

- double origin () const
- void setNeedle (QwtDialNeedle *)
- const QwtDialNeedle * needle () const
- QwtDialNeedle * needle ()
- · QRect boundingRect () const
- · QRect innerRect () const
- virtual QRect scaleInnerRect () const
- · virtual QSize sizeHint () const
- · virtual QSize minimumSizeHint () const
- void setScaleDraw (QwtRoundScaleDraw *)
- QwtRoundScaleDraw * scaleDraw ()
- const QwtRoundScaleDraw * scaleDraw () const

Protected Member Functions

- virtual void wheelEvent (QWheelEvent *)
- virtual void paintEvent (QPaintEvent *)
- virtual void changeEvent (QEvent *)
- virtual void drawFrame (QPainter *p)
- virtual void drawContents (QPainter *) const

Draw the contents inside the frame.

- virtual void drawFocusIndicator (QPainter *) const
- void invalidateCache ()
- virtual void drawScale (QPainter *, const QPointF ¢er, double radius) const
- virtual void drawScaleContents (QPainter *painter, const QPointF ¢er, double radius) const
- virtual void drawNeedle (QPainter *, const QPointF &, double radius, double direction, QPalette::ColorGroup)
 const
- virtual double scrolledTo (const QPoint &) const

Determine the value for a new position of the slider handle.

• virtual bool isScrollPosition (const QPoint &) const

Determine what to do when the user presses a mouse button.

· virtual void sliderChange ()

Calling update()

• virtual void scaleChange ()

Additional Inherited Members

12.27.1 Detailed Description

QwtDial class provides a rounded range control.

QwtDial is intended as base class for dial widgets like speedometers, compass widgets, clocks ...

A dial contains a scale and a needle indicating the current value of the dial. Depending on Mode one of them is fixed and the other is rotating. If not isReadOnly() the dial can be rotated by dragging the mouse or using keyboard inputs (see QwtAbstractSlider::keyPressEvent()). A dial might be wrapping, what means a rotation below/above one

limit continues on the other limit (f.e compass). The scale might cover any arc of the dial, its values are related to the origin() of the dial.

Often dials have to be updated very often according to values from external devices. For these high refresh rates QwtDial caches as much as possible. For derived classes it might be necessary to clear these caches manually according to attribute changes using invalidateCache().

See Also

QwtCompass, QwtAnalogClock, QwtDialNeedle

Note

The controls and dials examples shows different types of dials. QDial is more similar to QwtKnob than to QwtDial

12.27.2 Member Enumeration Documentation

12.27.2.1 enum QwtDial::Mode

Mode controlling whether the needle or the scale is rotating.

Enumerator

RotateNeedle The needle is rotating.

RotateScale The needle is fixed, the scales are rotating.

12.27.2.2 enum QwtDial::Shadow

Frame shadow.

Unfortunately it is not possible to use QFrame::Shadow as a property of a widget that is not derived from QFrame. The following enum is made for the designer only. It is safe to use QFrame::Shadow instead.

Enumerator

Plain QFrame::Plain.Raised QFrame::Raised.Sunken QFrame::Sunken.

12.27.3 Constructor & Destructor Documentation

12.27.3.1 QwtDial::QwtDial(QWidget * parent = NULL) [explicit]

Constructor.

Parameters

parent | Parent widget

Create a dial widget with no needle. The scale is initialized to [0.0, 360.0] and 360 steps (QwtAbstractSlider::set-TotalSteps()). The origin of the scale is at 90°,

The value is set to 0.0.

The default mode is QwtDial::RotateNeedle.

12.27.4 Member Function Documentation

12.27.4.1 QRect QwtDial::boundingRect () const

Returns

bounding rectangle of the dial including the frame

See Also

setLineWidth(), scaleInnerRect(), innerRect()

12.27.4.2 void QwtDial::changeEvent (QEvent * **event**) [protected], [virtual]

Change Event handler

Parameters

| event | Change event |
|-------|--------------|
| OVOIN | onango event |

Invalidates internal paint caches if necessary

12.27.4.3 void QwtDial::drawContents (QPainter * painter) const [protected], [virtual]

Draw the contents inside the frame.

QPalette::Window is the background color outside of the frame. QPalette::Base is the background color inside the frame. QPalette::WindowText is the background color inside the scale.

Parameters

| painter | Painter | 1 |
|---------|---------|-----|
| | | - 1 |

See Also

boundingRect(), innerRect(), scaleInnerRect(), QWidget::setPalette()

12.27.4.4 void QwtDial::drawFocusIndicator(QPainter * painter) const [protected], [virtual]

Draw the focus indicator

Parameters

| painter Painter |
|-------------------|
|-------------------|

12.27.4.5 void QwtDial::drawFrame (QPainter * painter) [protected], [virtual]

Draw the frame around the dial

Parameters

```
painter Painter
```

See Also

lineWidth(), frameShadow()

12.27.4.6 void QwtDial::drawNeedle (QPainter * painter, const QPointF & center, double radius, double direction, QPalette::ColorGroup colorGroup) const [protected], [virtual]

Draw the needle

Parameters

| painter | Painter |
|------------|---|
| center | Center of the dial |
| radius | Length for the needle |
| direction | Direction of the needle in degrees, counter clockwise |
| colorGroup | ColorGroup |

Reimplemented in QwtAnalogClock.

12.27.4.7 void QwtDial::drawScale (QPainter * *painter***, const QPointF &** *center***, double** *radius* **) const** [protected]**,** [virtual]

Draw the scale

Parameters

| painter | Painter |
|---------|---------------------|
| center | Center of the dial |
| radius | Radius of the scale |

12.27.4.8 void QwtDial::drawScaleContents (QPainter * painter, const QPointF & center, double radius) const [protected], [virtual]

Draw the contents inside the scale

Paints nothing.

Parameters

| painter | Painter |
|---------|-------------------------------|
| center | Center of the contents circle |
| radius | Radius of the contents circle |

Reimplemented in **QwtCompass**.

12.27.4.9 QwtDial::Shadow QwtDial::frameShadow () const

Returns

Frame shadow /sa setFrameShadow(), lineWidth(), QFrame::frameShadow()

12.27.4.10 QRect QwtDial::innerRect () const

Returns

bounding rectangle of the circle inside the frame

See Also

setLineWidth(), scaleInnerRect(), boundingRect()

12.27.4.11 void QwtDial::invalidateCache() [protected]

Invalidate the internal caches used to speed up repainting

12.27.4.12 bool QwtDial::isScrollPosition (const QPoint & pos) const [protected], [virtual]

Determine what to do when the user presses a mouse button.

```
Parameters
```

pos | Mouse position

Return values

True, when the inner circle contains pos

See Also

scrolledTo()

Implements QwtAbstractSlider.

12.27.4.13 int QwtDial::lineWidth () const

Returns

Line width of the frame

See Also

setLineWidth(), frameShadow(), lineWidth()

12.27.4.14 double QwtDial::maxScaleArc () const

Returns

Upper limit of the scale arc

See Also

setScaleArc()

12.27.4.15 QSize QwtDial::minimumSizeHint() const [virtual]

Returns

Minimum size hint

See Also

sizeHint()

12.27.4.16 double QwtDial::minScaleArc () const

Returns

Lower limit of the scale arc

See Also

setScaleArc()

12.27.4.17 QwtDial::Mode QwtDial::mode () const

Returns

Mode of the dial.

See Also

setMode(), origin(), setScaleArc(), value()

```
12.27.4.18 const QwtDialNeedle * QwtDial::needle ( ) const
Returns
     needle
See Also
     setNeedle()
12.27.4.19 QwtDialNeedle * QwtDial::needle ( )
Returns
     needle
See Also
     setNeedle()
12.27.4.20 double QwtDial::origin ( ) const
The origin is the angle where scale and needle is relative to.
Returns
     Origin of the dial
See Also
     setOrigin()
12.27.4.21 void QwtDial::paintEvent ( QPaintEvent * event ) [protected], [virtual]
Paint the dial
Parameters
                     Paint event
             event
12.27.4.22 void QwtDial::scaleChange() [protected], [virtual]
Invalidate the internal caches and call QwtAbstractSlider::scaleChange()
Reimplemented from QwtAbstractSlider.
12.27.4.23 QwtRoundScaleDraw * QwtDial::scaleDraw ( )
Returns
     the scale draw
12.27.4.24 const QwtRoundScaleDraw * QwtDial::scaleDraw ( ) const
Returns
     the scale draw
```

12.27.4.25 QRect QwtDial::scaleInnerRect() const [virtual]

Returns

rectangle inside the scale

See Also

setLineWidth(), boundingRect(), innerRect()

12.27.4.26 double QwtDial::scrolledTo (const QPoint & pos) const [protected], [virtual]

Determine the value for a new position of the slider handle.

Parameters

pos | Mouse position

Returns

Value for the mouse position

See Also

isScrollPosition()

Implements QwtAbstractSlider.

12.27.4.27 void QwtDial::setFrameShadow (Shadow shadow)

Sets the frame shadow value from the frame style.

Parameters

shadow Frame shadow

See Also

setLineWidth(), QFrame::setFrameShadow()

12.27.4.28 void QwtDial::setLineWidth (int lineWidth)

Sets the line width of the frame

Parameters

lineWidth Line width

See Also

setFrameShadow()

12.27.4.29 void QwtDial::setMaxScaleArc (double max)

Set the upper limit for the scale arc

Parameters

max Upper limit of the scale arc

See Also

setScaleArc(), setMinScaleArc()

12.27.4.30 void QwtDial::setMinScaleArc (double min)

Set the lower limit for the scale arc

Parameters

min Lower limit of the scale arc

See Also

setScaleArc(), setMaxScaleArc()

12.27.4.31 void QwtDial::setMode (Mode mode)

Change the mode of the dial.

Parameters

mode | New mode

In case of QwtDial::RotateNeedle the needle is rotating, in case of QwtDial::RotateScale, the needle points to origin() and the scale is rotating.

The default mode is QwtDial::RotateNeedle.

See Also

mode(), setValue(), setOrigin()

12.27.4.32 void QwtDial::setNeedle (QwtDialNeedle * needle)

Set a needle for the dial

Parameters

needle Needle

Warning

The needle will be deleted, when a different needle is set or in \sim QwtDial()

12.27.4.33 void QwtDial::setOrigin (double origin) [virtual]

Change the origin.

The origin is the angle where scale and needle is relative to.

Parameters

origin New origin

See Also

origin()

12.27.4.34 void QwtDial::setScaleArc (double minArc, double maxArc)

Change the arc of the scale

Parameters

| minArc | Lower limit |
|--------|-------------|
| maxArc | Upper limit |

See Also

minScaleArc(), maxScaleArc()

12.27.4.35 void QwtDial::setScaleDraw (QwtRoundScaleDraw * scaleDraw)

Set an individual scale draw

The motivation for setting a scale draw is often to overload QwtRoundScaleDraw::label() to return individual tick labels.

Parameters

| scaleDraw | Scale draw |
|-----------|------------|

Warning

The previous scale draw is deleted

12.27.4.36 QSize QwtDial::sizeHint() const [virtual]

Returns

Size hint

See Also

minimumSizeHint()

12.27.4.37 void QwtDial::wheelEvent (QWheelEvent * event) [protected], [virtual]

Wheel Event handler

Parameters

| event | Wheel event |
|-------|-------------|

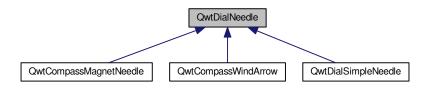
Reimplemented from QwtAbstractSlider.

12.28 QwtDialNeedle Class Reference

Base class for needles that can be used in a QwtDial.

#include <qwt_dial_needle.h>

 $Inheritance\ diagram\ for\ QwtDialNeedle:$



Public Member Functions

QwtDialNeedle ()

Constructor.

virtual ~QwtDialNeedle ()

Destructor.

- virtual void setPalette (const QPalette &)
- const QPalette & palette () const
- virtual void draw (QPainter *painter, const QPointF ¢er, double length, double direction, QPalette::Color-Group=QPalette::Active) const

Protected Member Functions

- virtual void drawNeedle (QPainter *painter, double length, QPalette::ColorGroup colorGroup) const =0
 Draw the needle.
- virtual void drawKnob (QPainter *, double width, const QBrush &, bool sunken) const
 Draw the knob.

12.28.1 Detailed Description

Base class for needles that can be used in a QwtDial.

QwtDialNeedle is a pointer that indicates a value by pointing to a specific direction.

See Also

QwtDial, QwtCompass

12.28.2 Member Function Documentation

12.28.2.1 void QwtDialNeedle::draw (QPainter * painter, const QPointF & center, double length, double direction, QPalette::ColorGroup = QPalette::Active) const [virtual]

Draw the needle

Parameters

| painter | Painter |
|------------|---|
| center | Center of the dial, start position for the needle |
| length | Length of the needle |
| direction | Direction of the needle, in degrees counter clockwise |
| colorGroup | Color group, used for painting |

12.28.2.2 virtual void QwtDialNeedle::drawNeedle (QPainter * painter, double length, QPalette::ColorGroup colorGroup)

const [protected], [pure virtual]

Draw the needle.

The origin of the needle is at position (0.0, 0.0) pointing in direction 0.0 (= east).

The painter is already initialized with translation and rotation.

Parameters

| painter | Painter |
|------------|--------------------------------|
| length | Length of the needle |
| colorGroup | Color group, used for painting |

See Also

setPalette(), palette()

Implemented in QwtCompassWindArrow, QwtCompassMagnetNeedle, and QwtDialSimpleNeedle.

12.28.2.3 const QPalette & QwtDialNeedle::palette () const

Returns

the palette of the needle.

12.28.2.4 void QwtDialNeedle::setPalette (const QPalette & palette) [virtual]

Sets the palette for the needle.

Parameters

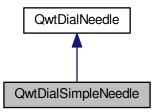
| palette | New Palette |
|---------|-------------|

12.29 QwtDialSimpleNeedle Class Reference

A needle for dial widgets.

#include <qwt_dial_needle.h>

Inheritance diagram for QwtDialSimpleNeedle:



Public Types

enum Style { Arrow, Ray }
 Style of the needle.

Public Member Functions

- QwtDialSimpleNeedle (Style, bool hasKnob=true, const QColor &mid=Qt::gray, const QColor &base=Qt::darkGray)
- void setWidth (double width)
- · double width () const

Protected Member Functions

• virtual void drawNeedle (QPainter *, double length, QPalette::ColorGroup) const

12.29.1 Detailed Description

A needle for dial widgets.

The following colors are used:

QPalette::Mid
 Pointer

QPalette::Base

Knob

See Also

QwtDial, QwtCompass

12.29.2 Member Enumeration Documentation

12.29.2.1 enum QwtDialSimpleNeedle::Style

Style of the needle.

Enumerator

Arrow Arrow.

Ray A straight line from the center.

12.29.3 Constructor & Destructor Documentation

12.29.3.1 QwtDialSimpleNeedle::QwtDialSimpleNeedle (Style style, bool hasKnob = true, const QColor & mid = Qt::gray, const QColor & base = Qt::darkGray)

Constructor

Parameters

| style | Style |
|---------|-------------------|
| hasKnob | With/Without knob |
| mid | Middle color |
| base | Base color |

12.29.4 Member Function Documentation

12.29.4.1 void QwtDialSimpleNeedle::drawNeedle (QPainter * painter, double length, QPalette::ColorGroup colorGroup)
const [protected], [virtual]

Draw the needle

Parameters

| | painter | Painter |
|---|------------|--------------------------------|
| | length | Length of the needle |
| ſ | colorGroup | Color group, used for painting |

Implements QwtDialNeedle.

12.29.4.2 void QwtDialSimpleNeedle::setWidth (double width)

Set the width of the needle

Parameters

| width | Width |
|-------|-------|
|-------|-------|

See Also

width()

12.29.4.3 double QwtDialSimpleNeedle::width () const

Returns

the width of the needle

See Also

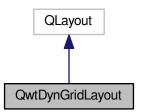
setWidth()

12.30 QwtDynGridLayout Class Reference

The QwtDynGridLayout class lays out widgets in a grid, adjusting the number of columns and rows to the current size.

#include <qwt_dyngrid_layout.h>

Inheritance diagram for QwtDynGridLayout:



Public Member Functions

- QwtDynGridLayout (QWidget *, int margin=0, int space=-1)
- QwtDynGridLayout (int space=-1)
- virtual \sim QwtDynGridLayout ()

Destructor.

virtual void invalidate ()

Invalidate all internal caches.

- void setMaxColumns (uint maxCols)
- uint maxColumns () const

Return the upper limit for the number of columns.

- uint numRows () const
- uint numColumns () const
- virtual void addltem (QLayoutItem *)

Add an item to the next free position.

- virtual QLayoutItem * itemAt (int index) const
- virtual QLayoutItem * takeAt (int index)
- virtual int count () const
- void setExpandingDirections (Qt::Orientations)
- · virtual Qt::Orientations expandingDirections () const

Returns whether this layout can make use of more space than sizeHint().

- QList< QRect > layoutItems (const QRect &, uint numCols) const
- · virtual int maxItemWidth () const
- virtual void setGeometry (const QRect &rect)
- virtual bool hasHeightForWidth () const
- virtual int heightForWidth (int) const
- · virtual QSize sizeHint () const
- virtual bool isEmpty () const
- uint itemCount () const
- · virtual uint columnsForWidth (int width) const

Calculate the number of columns for a given width.

Protected Member Functions

- void layoutGrid (uint numCols, QVector < int > &rowHeight, QVector < int > &colWidth) const
- void stretchGrid (const QRect &rect, uint numCols, QVector< int > &rowHeight, QVector< int > &colWidth)
 const

12.30.1 Detailed Description

The QwtDynGridLayout class lays out widgets in a grid, adjusting the number of columns and rows to the current size.

QwtDynGridLayout takes the space it gets, divides it up into rows and columns, and puts each of the widgets it manages into the correct cell(s). It lays out as many number of columns as possible (limited by maxColumns()).

12.30.2 Constructor & Destructor Documentation

12.30.2.1 QwtDynGridLayout::QwtDynGridLayout(QWidget*parent, int margin = 0, int spacing = -1) [explicit]

Parameters

| parent | Parent widget |
|---------|---------------|
| margin | Margin |
| spacing | Spacing |

12.30.2.2 QwtDynGridLayout::QwtDynGridLayout(int spacing = -1) [explicit]

Parameters

spacing Spacing

12.30.3 Member Function Documentation

12.30.3.1 void QwtDynGridLayout::addItem (QLayoutItem * item) [virtual]

Add an item to the next free position.

Parameters

item Layout item

12.30.3.2 uint QwtDynGridLayout::columnsForWidth(int width)const [virtual]

Calculate the number of columns for a given width.

The calculation tries to use as many columns as possible (limited by maxColumns())

Parameters

width Available width for all columns

Returns

Number of columns for a given width

See Also

maxColumns(), setMaxColumns()

12.30.3.3 int QwtDynGridLayout::count() const [virtual]

Returns

Number of items in the layout

12.30.3.4 Qt::Orientations QwtDynGridLayout::expandingDirections () const [virtual]

Returns whether this layout can make use of more space than sizeHint().

A value of Qt::Vertical or Qt::Horizontal means that it wants to grow in only one dimension, while Qt::Vertical | Qt::Horizontal means that it wants to grow in both dimensions.

Returns

Orientations, where the layout expands

See Also

setExpandingDirections()

12.30.3.5 bool QwtDynGridLayout::hasHeightForWidth()const [virtual]

Returns

true: QwtDynGridLayout implements heightForWidth().

See Also

heightForWidth()

12.30.3.6 int QwtDynGridLayout::heightForWidth (int width) const [virtual]

Returns

The preferred height for this layout, given a width.

See Also

hasHeightForWidth()

12.30.3.7 bool QwtDynGridLayout::isEmpty() const [virtual]

Returns

true if this layout is empty.

12.30.3.8 QLayoutItem * QwtDynGridLayout::itemAt (int index) const [virtual]

Find the item at a specific index

Parameters

| index | Index |
|-------|-------|
|-------|-------|

Returns

Item at a specific index

See Also

takeAt()

12.30.3.9 uint QwtDynGridLayout::itemCount () const

Returns

number of layout items

12.30.3.10 void QwtDynGridLayout::layoutGrid (uint numColumns, QVector< int > & rowHeight, QVector< int > & colWidth) const [protected]

Calculate the dimensions for the columns and rows for a grid of numColumns columns.

Parameters

| numColumns | Number of columns. |
|------------|--|
| rowHeight | Array where to fill in the calculated row heights. |
| colWidth | Array where to fill in the calculated column widths. |

12.30.3.11 QList < QRect > QwtDynGridLayout::layoutItems (const QRect & rect, uint numColumns) const

Calculate the geometries of the layout items for a layout with numColumns columns and a given rectangle.

Parameters

| rect Rect where to place the items |
|------------------------------------|
|------------------------------------|

numColumns Number of columns Returns item geometries 12.30.3.12 uint QwtDynGridLayout::maxColumns () const Return the upper limit for the number of columns. 0 means unlimited, what is the default. Returns Upper limit for the number of columns See Also setMaxColumns() 12.30.3.13 int QwtDynGridLayout::maxItemWidth() const [virtual] Returns the maximum width of all layout items 12.30.3.14 uint QwtDynGridLayout::numColumns () const Returns Number of columns of the current layout. See Also numRows() Warning The number of columns might change whenever the geometry changes 12.30.3.15 uint QwtDynGridLayout::numRows () const Returns Number of rows of the current layout. See Also numColumns() Warning The number of rows might change whenever the geometry changes

Set whether this layout can make use of more space than sizeHint(). A value of Qt::Vertical or Qt::Horizontal means that it wants to grow in only one dimension, while Qt::Vertical | Qt::Horizontal means that it wants to grow in both dimensions. The default value is 0.

12.30.3.16 void QwtDynGridLayout::setExpandingDirections (Qt::Orientations expanding)

Parameters

| expanding | Or'd orientations | 7 |
|-----------|-------------------|---|
|-----------|-------------------|---|

See Also

expandingDirections()

12.30.3.17 void QwtDynGridLayout::setGeometry (const QRect & rect) [virtual]

Reorganizes columns and rows and resizes managed items within a rectangle.

Parameters

| rect | Layout geometry |
|------|-----------------|

12.30.3.18 void QwtDynGridLayout::setMaxColumns (uint maxColumns)

Limit the number of columns.

Parameters

| maxColumns | upper limit, 0 means unlimited |
|------------|--------------------------------|

See Also

maxColumns()

12.30.3.19 QSize QwtDynGridLayout::sizeHint()const [virtual]

Return the size hint. If maxColumns() > 0 it is the size for a grid with maxColumns() columns, otherwise it is the size for a grid with only one row.

Returns

Size hint

See Also

maxColumns(), setMaxColumns()

12.30.3.20 void QwtDynGridLayout::stretchGrid (const QRect & rect, uint numColumns, QVector < int > & rowHeight, QVector < int > & colWidth) const [protected]

Stretch columns in case of expanding() & QSizePolicy::Horizontal and rows in case of expanding() & QSizePolicy::Vertical to fill the entire rect. Rows and columns are stretched with the same factor.

Parameters

| rect | Bounding rectangle |
|------------|--|
| numColumns | Number of columns |
| rowHeight | Array to be filled with the calculated row heights |
| colWidth | Array to be filled with the calculated column widths |

See Also

setExpanding(), expanding()

12.30.3.21 QLayoutItem * QwtDynGridLayout::takeAt (int index) [virtual]

Find the item at a specific index and remove it from the layout

Parameters

| index | Index |
|-------|-------|

Returns

Layout item, removed from the layout

See Also

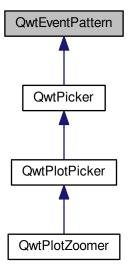
itemAt()

12.31 QwtEventPattern Class Reference

A collection of event patterns.

#include <qwt_event_pattern.h>

Inheritance diagram for QwtEventPattern:



Classes

• class KeyPattern

A pattern for key events.

• class MousePattern

A pattern for mouse events.

Public Types

enum MousePatternCode {
 MouseSelect1, MouseSelect2, MouseSelect3, MouseSelect4,
 MouseSelect5, MouseSelect6, MousePatternCount }

Symbolic mouse input codes.

enum KeyPatternCode {
 KeySelect1, KeySelect2, KeyAbort, KeyLeft,
 KeyRight, KeyUp, KeyDown, KeyRedo,
 KeyUndo, KeyHome, KeyPatternCount }

Symbolic keyboard input codes.

Public Member Functions

- QwtEventPattern ()
- virtual ~QwtEventPattern ()

Destructor.

- void initMousePattern (int numButtons)
- void initKeyPattern ()
- void setMousePattern (MousePatternCode, Qt::MouseButton button, Qt::KeyboardModifiers=Qt::NoModifier)
- void setKeyPattern (KeyPatternCode, int keyCode, Qt::KeyboardModifiers modifierCodes=Qt::NoModifier)
- void setMousePattern (const QVector< MousePattern > &)

Change the mouse event patterns.

void setKeyPattern (const QVector< KeyPattern > &)

Change the key event patterns.

- const QVector< MousePattern > & mousePattern () const
- const QVector< KeyPattern > & keyPattern () const
- QVector< MousePattern > & mousePattern ()
- QVector< KeyPattern > & keyPattern ()
- bool mouseMatch (MousePatternCode, const QMouseEvent *) const

Compare a mouse event with an event pattern.

• bool keyMatch (KeyPatternCode, const QKeyEvent *) const

Compare a key event with an event pattern.

Protected Member Functions

virtual bool mouseMatch (const MousePattern &, const QMouseEvent *) const
 Compare a mouse event with an event pattern.

virtual bool keyMatch (const KeyPattern &, const QKeyEvent *) const

Compare a key event with an event pattern.

12.31.1 Detailed Description

A collection of event patterns.

QwtEventPattern introduces an level of indirection for mouse and keyboard inputs. Those are represented by symbolic names, so the application code can be configured by individual mappings.

See Also

QwtPicker, QwtPickerMachine, QwtPlotZoomer

12.31.2 Member Enumeration Documentation

12.31.2.1 enum QwtEventPattern::KeyPatternCode

Symbolic keyboard input codes.

Individual settings can be configured using setKeyPattern()

See Also

setKeyPattern(), setMousePattern()

Enumerator

KeySelect1 Qt::Key_Return.
KeySelect2 Qt::Key_Space.
KeyAbort Qt::Key_Escape.
KeyLeft Qt::Key_Left.
KeyRight Qt::Key_Right.
KeyUp Qt::Key_Up.
KeyDown Qt::Key_Down.
KeyRedo Qt::Key_Plus.
KeyUndo Qt::Key_Minus.
KeyHome Qt::Key_Escape.
KeyPatternCount Number of key patterns.

12.31.2.2 enum QwtEventPattern::MousePatternCode

Symbolic mouse input codes.

QwtEventPattern implements 3 different settings for mice with 1, 2, or 3 buttons that can be activated using init-MousePattern(). The default setting is for 3 button mice.

Individual settings can be configured using setMousePattern().

See Also

initMousePattern(), setMousePattern(), setKeyPattern()

Enumerator

MouseSelect1 The default setting for 1, 2 and 3 button mice is:

- · Qt::LeftButton
- · Qt::LeftButton
- · Qt::LeftButton

MouseSelect2 The default setting for 1, 2 and 3 button mice is:

- Qt::LeftButton + Qt::ControlModifier
- · Qt::RightButton
- Qt::RightButton

MouseSelect3 The default setting for 1, 2 and 3 button mice is:

- Qt::LeftButton + Qt::AltModifier
- · Qt::LeftButton + Qt::AltModifier
- · Qt::MidButton

MouseSelect4 The default setting for 1, 2 and 3 button mice is:

- Qt::LeftButton + Qt::ShiftModifier
- Qt::LeftButton + Qt::ShiftModifier
- Qt::LeftButton + Qt::ShiftModifier

MouseSelect5 The default setting for 1, 2 and 3 button mice is:

- Qt::LeftButton + Qt::ControlButton | Qt::ShiftModifier
- Qt::RightButton + Qt::ShiftModifier

• Qt::RightButton + Qt::ShiftModifier

MouseSelect6 The default setting for 1, 2 and 3 button mice is:

- Qt::LeftButton + Qt::AltModifier + Qt::ShiftModifier
- Qt::LeftButton + Qt::AltModifier | Qt::ShiftModifier
- Qt::MidButton + Qt::ShiftModifier

MousePatternCount Number of mouse patterns.

12.31.3 Constructor & Destructor Documentation

12.31.3.1 QwtEventPattern::QwtEventPattern ()

Constructor

See Also

MousePatternCode, KeyPatternCode

12.31.4 Member Function Documentation

12.31.4.1 void QwtEventPattern::initKeyPattern ()

Set default mouse patterns.

See Also

KeyPatternCode

12.31.4.2 void QwtEventPattern::initMousePattern (int numButtons)

Set default mouse patterns, depending on the number of mouse buttons

Parameters

| numButtons | Number of mouse buttons (<= 3) |
|------------|----------------------------------|
|------------|----------------------------------|

See Also

MousePatternCode

12.31.4.3 bool QwtEventPattern::keyMatch (KeyPatternCode code, const QKeyEvent * event) const

Compare a key event with an event pattern.

A key event matches the pattern when both have the same key value and in the state value the same key flags (Qt::KeyButtonMask) are set.

Parameters

| code | Index of the event pattern |
|-------|----------------------------|
| event | Key event |

Returns

true if matches

See Also

mouseMatch()

12.31.4.4 bool QwtEventPattern::keyMatch (const KeyPattern & pattern, const QKeyEvent * event) const [protected], [virtual]

Compare a key event with an event pattern.

A key event matches the pattern when both have the same key value and in the state value the same key flags (Qt::KeyButtonMask) are set.

Parameters

| pattern | Key event pattern |
|---------|-------------------|
| event | Key event |

Returns

true if matches

See Also

mouseMatch()

12.31.4.5 const QVector < QwtEventPattern::KeyPattern > & QwtEventPattern::keyPattern () const

Returns

Key pattern

12.31.4.6 QVector < QwtEventPattern::KeyPattern > & QwtEventPattern::keyPattern ()

Returns

Key pattern

12.31.4.7 bool QwtEventPattern::mouseMatch (MousePatternCode code, const QMouseEvent * event) const

Compare a mouse event with an event pattern.

A mouse event matches the pattern when both have the same button value and in the state value the same key flags(Qt::KeyButtonMask) are set.

Parameters

| code | Index of the event pattern |
|-------|----------------------------|
| event | Mouse event |

Returns

true if matches

See Also

keyMatch()

12.31.4.8 bool QwtEventPattern::mouseMatch (const MousePattern & pattern, const QMouseEvent * event) const [protected], [virtual]

Compare a mouse event with an event pattern.

A mouse event matches the pattern when both have the same button value and in the state value the same key flags(Qt::KeyButtonMask) are set.

Parameters

| pattern | Mouse event pattern |
|---------|---------------------|
| event | Mouse event |

Returns

true if matches

See Also

keyMatch()

12.31.4.9 const QVector < QwtEventPattern::MousePattern > & QwtEventPattern::mousePattern () const

Returns

Mouse pattern

12.31.4.10 QVector < QwtEventPattern::MousePattern > & QwtEventPattern::mousePattern ()

Returns

Mouse pattern

12.31.4.11 void QwtEventPattern::setKeyPattern (KeyPatternCode pattern, int key, Qt::KeyboardModifiers modifiers = Qt::NoModifier)

Change one key pattern

Parameters

| pattern | Index of the pattern |
|-----------|----------------------|
| key | Key |
| modifiers | Keyboard modifiers |

See Also

QKeyEvent

12.31.4.12 void QwtEventPattern::setMousePattern (MousePatternCode pattern, Qt::MouseButton button, Qt::KeyboardModifiers modifiers = Qt::NoModifier)

Change one mouse pattern

Parameters

| pattern | Index of the pattern |
|-----------|----------------------|
| button | Button |
| modifiers | Keyboard modifiers |

See Also

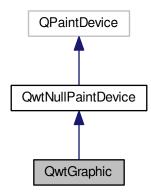
QMouseEvent

12.32 QwtGraphic Class Reference

A paint device for scalable graphics.

#include <qwt_graphic.h>

Inheritance diagram for QwtGraphic:



Public Types

- enum RenderHint { RenderPensUnscaled = 0x1 }
- typedef QFlags< RenderHint > RenderHints
 Render hints.

Public Member Functions

• QwtGraphic ()

Constructor.

• QwtGraphic (const QwtGraphic &)

Copy constructor.

virtual ~QwtGraphic ()

Destructor.

• QwtGraphic & operator= (const QwtGraphic &)

Assignment operator.

• void reset ()

Clear all stored commands.

- bool isNull () const
- bool isEmpty () const
- void render (QPainter *) const

Replay all recorded painter commands.

- void render (QPainter *, const QSizeF &, Qt::AspectRatioMode=Qt::IgnoreAspectRatio) const Replay all recorded painter commands.
- void render (QPainter *, const QRectF &, Qt::AspectRatioMode=Qt::IgnoreAspectRatio) const Replay all recorded painter commands.
- void render (QPainter *, const QPointF &, Qt::Alignment=Qt::AlignTop|Qt::AlignLeft) const Replay all recorded painter commands.
- QPixmap toPixmap () const

Convert the graphic to a QPixmap.

• QPixmap toPixmap (const QSize &, Qt::AspectRatioMode=Qt::IgnoreAspectRatio) const

Convert the graphic to a QPixmap.

• QImage tolmage () const

Convert the graphic to a QImage.

QImage toImage (const QSize &, Qt::AspectRatioMode=Qt::IgnoreAspectRatio) const

Convert the graphic to a QImage.

QRectF scaledBoundingRect (double sx, double sy) const

Calculate the target rectangle for scaling the graphic.

- QRectF boundingRect () const
- QRectF controlPointRect () const
- · const QVector
 - < QwtPainterCommand > & commands () const
- void setCommands (QVector < QwtPainterCommand > &)

Append paint commands.

• void setDefaultSize (const QSizeF &)

Set a default size.

· QSizeF defaultSize () const

Default size.

- void setRenderHint (RenderHint, bool on=true)
- · bool testRenderHint (RenderHint) const

Protected Member Functions

- virtual QSize sizeMetrics () const
- virtual void drawPath (const QPainterPath &)
- virtual void drawPixmap (const QRectF &, const QPixmap &, const QRectF &)

Store a pixmap command in the command list.

• virtual void drawlmage (const QRectF &, const QImage &, const QRectF &, Qt::ImageConversionFlags)

Store a image command in the command list.

virtual void updateState (const QPaintEngineState &state)

Store a state command in the command list.

12.32.1 Detailed Description

A paint device for scalable graphics.

QwtGraphic is the representation of a graphic that is tailored for scalability. Like QPicture it will be initialized by QPainter operations and can be replayed later to any target paint device.

While the usual image representations Qlmage and QPixmap are not scalable Qt offers two paint devices, that might be candidates for representing a vector graphic:

QPicture

Unfortunately QPicture had been forgotten, when Qt4 introduced floating point based render engines. Its API is still on integers, what make it unusable for proper scaling.

· QSvgRenderer/QSvgGenerator

Unfortunately QSvgRenderer hides to much information about its nodes in internal APIs, that are necessary for proper layout calculations. Also it is derived from QObject and can't be copied like QImage/QPixmap.

QwtGraphic maps all scalable drawing primitives to a QPainterPath and stores them together with the painter state changes (pen, brush, transformation ...) in a list of QwtPaintCommands. For being a complete QPaintDevice it also stores pixmaps or images, what is somehow against the idea of the class, because these objects can't be scaled without a loss in quality.

The main issue about scaling a QwtGraphic object are the pens used for drawing the outlines of the painter paths. While non cosmetic pens (QPen::isCosmetic()) are scaled with the same ratio as the path, cosmetic pens have a fixed width. A graphic might have paths with different pens - cosmetic and non-cosmetic.

QwtGraphic caches 2 different rectangles:

· control point rectangle

The control point rectangle is the bounding rectangle of all control point rectangles of the painter paths, or the target rectangle of the pixmaps/images.

· bounding rectangle

The bounding rectangle extends the control point rectangle by what is needed for rendering the outline with an unscaled pen.

Because the offset for drawing the outline depends on the shape of the painter path (the peak of a triangle is different than the flat side) scaling with a fixed aspect ratio always needs to be calculated from the control point rectangle.

See Also

QwtPainterCommand

12.32.2 Member Typedef Documentation

12.32.2.1 typedef QFlags<RenderHint> QwtGraphic::RenderHints

Render hints.

The default setting is to disable all hints

12.32.3 Member Enumeration Documentation

12.32.3.1 enum QwtGraphic::RenderHint

Hint how to render a graphic

See Also

setRenderHint(), testRenderHint()

Enumerator

RenderPensUnscaled When rendering a QwtGraphic a specific scaling between the controlPointRect() and the coordinates of the target rectangle is set up internally in render().

When RenderPensUnscaled is set this specific scaling is applied for the control points only, but not for the pens. All other painter transformations (set up by application code) are supposed to work like usual. See Also

render();

12.32.4 Constructor & Destructor Documentation

12.32.4.1 QwtGraphic::QwtGraphic ()

Constructor.

Initializes a null graphic

See Also

isNull()

12.32.4.2 QwtGraphic::QwtGraphic (const QwtGraphic & other)

Copy constructor.

Parameters

other Source

See Also

operator=()

12.32.5 Member Function Documentation

12.32.5.1 QRectF QwtGraphic::boundingRect () const

The bounding rectangle is the controlPointRect() extended by the areas needed for rendering the outlines with unscaled pens.

Returns

Bounding rectangle of the graphic

See Also

controlPointRect(), scaledBoundingRect()

12.32.5.2 const QVector < QwtPainterCommand > & QwtGraphic::commands () const

Returns

List of recorded paint commands

See Also

setCommands()

12.32.5.3 QRectF QwtGraphic::controlPointRect () const

The control point rectangle is the bounding rectangle of all control points of the paths and the target rectangles of the images/pixmaps.

Returns

Control point rectangle

See Also

boundingRect(), scaledBoundingRect()

12.32.5.4 QSizeF QwtGraphic::defaultSize () const

Default size.

When a non empty size has been assigned by setDefaultSize() this size will be returned. Otherwise the default size is the size of the bounding rectangle.

The default size is used in all methods rendering the graphic, where no size is explicitly specified.

Returns

Default size

See Also

setDefaultSize(), boundingRect()

12.32.5.5 void QwtGraphic::drawImage (const QRectF & rect, const QImage & image, const QRectF & subRect, Qt::ImageConversionFlags flags) [protected], [virtual]

Store a image command in the command list.

Parameters

| rect | traget rectangle |
|---------|--|
| image | Image to be painted |
| subRect | Reactangle of the pixmap to be painted |
| flags | Image conversion flags |

See Also

QPaintEngine::drawImage()

Reimplemented from QwtNullPaintDevice.

12.32.5.6 void QwtGraphic::drawPath (const QPainterPath & path) [protected], [virtual]

Store a path command in the command list

Parameters

| path | Painter path |
|------|--------------|

See Also

QPaintEngine::drawPath()

Reimplemented from QwtNullPaintDevice.

12.32.5.7 void QwtGraphic::drawPixmap (const QRectF & rect, const QPixmap & pixmap, const QRectF & subRect)
[protected], [virtual]

Store a pixmap command in the command list.

Parameters

| rect | target rectangle |
|---------|--|
| pixmap | Pixmap to be painted |
| subRect | Reactangle of the pixmap to be painted |

See Also

QPaintEngine::drawPixmap()

Reimplemented from QwtNullPaintDevice.

12.32.5.8 bool QwtGraphic::isEmpty () const

Returns

True, when the bounding rectangle is empty

See Also

boundingRect(), isNull()

12.32.5.9 bool QwtGraphic::isNull () const

Returns

True, when no painter commands have been stored

See Also

isEmpty(), commands()

12.32.5.10 QwtGraphic & QwtGraphic::operator= (const QwtGraphic & other)
Assignment operator.

Parameters

| other | Source |
|-------|--------|
|-------|--------|

Returns

A reference of this object

12.32.5.11 void QwtGraphic::render (QPainter * painter) const

Replay all recorded painter commands.

Parameters

| painter | Qt painter |
|---------|------------|

12.32.5.12 void QwtGraphic::render (QPainter * painter, const QSizeF & size, Qt::AspectRatioMode aspectRatioMode = Qt::IgnoreAspectRatio) const

Replay all recorded painter commands.

The graphic is scaled to fit into the rectangle of the given size starting at ($0,\,0$).

Parameters

| painter | Qt painter |
|--------------|---|
| size | Size for the scaled graphic |
| aspectRatio- | Mode how to scale - See Qt::AspectRatioMode |
| Mode | |

12.32.5.13 void QwtGraphic::render (QPainter * painter, const QRectF & rect, Qt::AspectRatioMode aspectRatioMode = Qt::IgnoreAspectRatio) const

Replay all recorded painter commands.

The graphic is scaled to fit into the given rectangle

Parameters

| painter | Qt painter |
|--------------|---|
| rect | Rectangle for the scaled graphic |
| aspectRatio- | Mode how to scale - See Qt::AspectRatioMode |
| Mode | |

12.32.5.14 void QwtGraphic::render (QPainter * painter, const QPointF & pos, Qt::Alignment alignment = Qt::AlignTop | Qt::AlignLeft) const

Replay all recorded painter commands.

The graphic is scaled to the defaultSize() and aligned to a position.

Parameters

| painter | Qt painter |
|-----------|---|
| pos | Reference point, where to render |
| alignment | Flags how to align the target rectangle to pos. |

12.32.5.15 void QwtGraphic::reset ()

Clear all stored commands.

See Also

isNull()

12.32.5.16 QRectF QwtGraphic::scaledBoundingRect (double sx, double sy) const

Calculate the target rectangle for scaling the graphic.

Parameters

| SX | Horizontal scaling factor |
|----|---------------------------|
| sy | Vertical scaling factor |

Note

In case of paths that are painted with a cosmetic pen (see QPen::isCosmetic()) the target rectangle is different to multiplying the bounding rectangle.

Returns

Scaled bounding rectangle

See Also

boundingRect(), controlPointRect()

12.32.5.17 void QwtGraphic::setCommands (QVector < QwtPainterCommand > & commands)

Append paint commands.

Parameters

| commands | Paint commands |
|----------|----------------|

See Also

commands()

12.32.5.18 void QwtGraphic::setDefaultSize (const QSizeF & size)

Set a default size.

The default size is used in all methods rendering the graphic, where no size is explicitly specified. Assigning an empty size means, that the default size will be calculated from the bounding rectangle.

The default setting is an empty size.

Parameters

| size | Default size |
|------|--------------|

See Also

defaultSize(), boundingRect()

12.32.5.19 void QwtGraphic::setRenderHint (RenderHint hint, bool on = true)

Toggle an render hint

Parameters

| hint | Render hint |
|------|-------------|
| on | true/false |

See Also

testRenderHint(), RenderHint

12.32.5.20 QSize QwtGraphic::sizeMetrics () const [protected], [virtual]

Returns

Ceiled defaultSize()

Implements QwtNullPaintDevice.

12.32.5.21 bool QwtGraphic::testRenderHint (RenderHint hint) const

Test a render hint

Parameters

| hint | Render hint |
|------|-------------|
|------|-------------|

Returns

true/false

See Also

setRenderHint(), RenderHint

12.32.5.22 Qlmage QwtGraphic::tolmage () const

Convert the graphic to a QImage.

All pixels of the image get initialized by 0 (transparent) before the graphic is scaled and rendered on it.

The format of the image is QImage::Format_ARGB32_Premultiplied.

The size of the image is the default size (ceiled to integers) of the graphic.

Returns

The graphic as image in default size

See Also

defaultSize(), toPixmap(), render()

12.32.5.23 Qlmage QwtGraphic::tolmage (const QSize & size, Qt::AspectRatioMode aspectRatioMode = Qt::IgnoreAspectRatio) const

Convert the graphic to a QImage.

All pixels of the image get initialized by 0 (transparent) before the graphic is scaled and rendered on it.

The format of the image is QImage::Format_ARGB32_Premultiplied.

Parameters

| size | Size of the image |
|--------------|---------------------------------------|
| aspectRatio- | Aspect ratio how to scale the graphic |
| Mode | |

Returns

The graphic as image

See Also

toPixmap(), render()

12.32.5.24 QPixmap QwtGraphic::toPixmap () const

Convert the graphic to a QPixmap.

All pixels of the pixmap get initialized by Qt::transparent before the graphic is scaled and rendered on it.

The size of the pixmap is the default size (ceiled to integers) of the graphic.

Returns

The graphic as pixmap in default size

See Also

defaultSize(), tolmage(), render()

12.32.5.25 QPixmap QwtGraphic::toPixmap (const QSize & size, Qt::AspectRatioMode aspectRatioMode = Qt::IgnoreAspectRatio) const

Convert the graphic to a QPixmap.

All pixels of the pixmap get initialized by Qt::transparent before the graphic is scaled and rendered on it.

Parameters

| size | Size of the image |
|--------------|---------------------------------------|
| aspectRatio- | Aspect ratio how to scale the graphic |
| Mode | |

Returns

The graphic as pixmap

See Also

tolmage(), render()

12.32.5.26 void QwtGraphic::updateState (const QPaintEngineState & state) [protected], [virtual]

Store a state command in the command list.

Parameters

state State to be stored

See Also

QPaintEngine::updateState()

Reimplemented from QwtNullPaintDevice.

12.33 QwtInterval Class Reference

A class representing an interval.

```
#include <qwt_interval.h>
```

Public Types

- enum BorderFlag { IncludeBorders = 0x00, ExcludeMinimum = 0x01, ExcludeMaximum = 0x02, Exclude-Borders = ExcludeMinimum | ExcludeMaximum }
- typedef QFlags < BorderFlag > BorderFlags
 Border flags.

Public Member Functions

· QwtInterval ()

Default Constructor.

- QwtInterval (double minValue, double maxValue, BorderFlags=IncludeBorders)
- void setInterval (double minValue, double maxValue, BorderFlags=IncludeBorders)
- · QwtInterval normalized () const

Normalize the limits of the interval.

- · QwtInterval inverted () const
- · QwtInterval limited (double minValue, double maxValue) const
- bool operator== (const QwtInterval &) const

Compare two intervals.

• bool operator!= (const QwtInterval &) const

Compare two intervals.

- void setBorderFlags (BorderFlags)
- · BorderFlags borderFlags () const
- double minValue () const
- double maxValue () const
- · double width () const

Return the width of an interval.

- void setMinValue (double)
- void setMaxValue (double)
- · bool contains (double value) const
- bool intersects (const QwtInterval &) const

Test if two intervals overlap.

• QwtInterval intersect (const QwtInterval &) const

Intersect 2 intervals.

· QwtInterval unite (const QwtInterval &) const

Unite 2 intervals.

• QwtInterval operator (const QwtInterval &) const

QwtInterval operator& (const QwtInterval &) const

Intersection of two intervals.

QwtInterval & operator = (const QwtInterval &)

Unite this interval with the given interval.

QwtInterval & operator&= (const QwtInterval &)

Intersect this interval with the given interval.

QwtInterval extend (double value) const

Extend the interval.

- QwtInterval operator (double) const
- QwtInterval & operator = (double)
- bool isValid () const
- bool isNull () const
- void invalidate ()
- QwtInterval symmetrize (double value) const

12.33.1 Detailed Description

A class representing an interval.

The interval is represented by 2 doubles, the lower and the upper limit.

12.33.2 Member Enumeration Documentation

12.33.2.1 enum QwtInterval::BorderFlag

Flag indicating if a border is included or excluded

See Also

```
setBorderFlags(), borderFlags()
```

Enumerator

IncludeBorders Min/Max values are inside the interval.

ExcludeMinimum Min value is not included in the interval.

ExcludeMaximum Max value is not included in the interval.

ExcludeBorders Min/Max values are not included in the interval.

12.33.3 Constructor & Destructor Documentation

```
12.33.3.1 QwtInterval::QwtInterval() [inline]
```

Default Constructor.

Creates an invalid interval [0.0, -1.0]

See Also

setInterval(), isValid()

12.33.3.2 QwtInterval::QwtInterval (double minValue, double maxValue, BorderFlags borderFlags = IncludeBorders) [inline]

Constructor

Build an interval with from min/max values

Parameters

| minValue | Minimum value |
|-------------|-------------------------|
| maxValue | Maximum value |
| borderFlags | Include/Exclude borders |

12.33.4 Member Function Documentation

12.33.4.1 QwtInterval::BorderFlags QwtInterval::borderFlags () const [inline]

Returns

Border flags

See Also

setBorderFlags()

12.33.4.2 bool QwtInterval::contains (double value) const

Test if a value is inside an interval

Parameters

| _ | | |
|---|-------|---------|
| | , | |
| | Value | I Value |
| | value | value |
| | | |

Returns

true, if value >= minValue() && value <= maxValue()

12.33.4.3 QwtInterval QwtInterval::extend (double value) const

Extend the interval.

If value is below minValue(), value becomes the lower limit. If value is above maxValue(), value becomes the upper limit.

extend() has no effect for invalid intervals

Parameters

| value | Value |
|-------|-------|
| value | value |

Returns

extended interval

See Also

isValid()

12.33.4.4 QwtInterval QwtInterval::intersect (const QwtInterval & other) const

Intersect 2 intervals.

Parameters

other Interval to be intersect with

Returns

Intersection

12.33.4.5 bool QwtInterval::intersects (const QwtInterval & other) const

Test if two intervals overlap.

Parameters

```
other Interval
```

Returns

True, when the intervals are intersecting

12.33.4.6 void QwtInterval::invalidate() [inline]

Invalidate the interval

The limits are set to interval [0.0, -1.0]

See Also

isValid()

12.33.4.7 QwtInterval QwtInterval::inverted () const

Invert the limits of the interval

Returns

Inverted interval

See Also

normalized()

12.33.4.8 bool QwtInterval::isNull () const [inline]

Returns

true, if isValid() && (minValue() >= maxValue())

12.33.4.9 bool QwtInterval::isValid () const [inline]

A interval is valid when minValue() <= maxValue(). In case of QwtInterval::ExcludeBorders it is true when minValue() < maxValue()

Returns

True, when the interval is valid

12.33.4.10 QwtInterval QwtInterval::limited (double lowerBound, double upperBound) const

Limit the interval, keeping the border modes

Parameters

| IowerBound | Lower limit |
|------------|-------------|
| upperBound | Upper limit |

Returns

Limited interval

12.33.4.11 double QwtInterval::maxValue () const [inline]

Returns

Upper limit of the interval

12.33.4.12 double QwtInterval::minValue () const [inline]

Returns

Lower limit of the interval

12.33.4.13 QwtInterval QwtInterval::normalized () const

Normalize the limits of the interval.

If maxValue() < minValue() the limits will be inverted.

Returns

Normalized interval

See Also

isValid(), inverted()

12.33.4.14 bool QwtInterval::operator!= (const QwtInterval & other) const [inline]

Compare two intervals.

Parameters

| other | Interval to compare with |
|-------|--------------------------|

Returns

True, when this and other are not equal

12.33.4.15 QwtInterval QwtInterval::operator& (const QwtInterval & other) const [inline]

Intersection of two intervals.

Parameters

| other | Interval to intersect with |
|-------|----------------------------|
|-------|----------------------------|

Returns

Intersection of this and other

See Also

intersect()

12.33.4.16 QwtInterval & QwtInterval::operator&= (const QwtInterval & other)

Intersect this interval with the given interval.

Parameters

other Interval to be intersected with

Returns

This interval

12.33.4.17 bool QwtInterval::operator== (const QwtInterval & other) const [inline]

Compare two intervals.

Parameters

other Interval to compare with

Returns

True, when this and other are equal

12.33.4.18 QwtInterval QwtInterval::operator (const QwtInterval & other) const [inline]

Union of two intervals

Parameters

other Interval to unite with

Returns

Union of this and other

See Also

unite()

12.33.4.19 QwtInterval QwtInterval::operator (double value) const [inline]

Extend an interval

Parameters

value Value

Returns

Extended interval

See Also

extend()

12.33.4.20 QwtInterval & QwtInterval::operator = (const QwtInterval & other)

Unite this interval with the given interval.

Parameters

| other | Interval to be united with |
|-------|----------------------------|
|-------|----------------------------|

Returns

This interval

12.33.4.21 QwtInterval & QwtInterval::operator = (double value)

Extend an interval

Parameters

| value | Value |
|-------|-------|

Returns

Reference of the extended interval

See Also

extend()

12.33.4.22 void QwtInterval::setBorderFlags (BorderFlags borderFlags) [inline]

Change the border flags

Parameters

| borderFlags Or'd BorderMode flags |
|-------------------------------------|
|-------------------------------------|

See Also

borderFlags()

12.33.4.23 void QwtInterval::setInterval (double minValue, double maxValue, BorderFlags borderFlags = IncludeBorders) [inline]

Assign the limits of the interval

Parameters

| ſ | minValue | Minimum value |
|---|-------------|-------------------------|
| ſ | maxValue | Maximum value |
| Ī | borderFlags | Include/Exclude borders |

12.33.4.24 void QwtInterval::setMaxValue (double maxValue) [inline]

Assign the upper limit of the interval

Parameters

| maxValue | Maximum value |
|----------|---------------|

12.33.4.25 void QwtInterval::setMinValue (double minValue) [inline]

Assign the lower limit of the interval

Parameters

| minValue | Minimum value | 1 |
|----------|---------------|---|
|----------|---------------|---|

12.33.4.26 QwtInterval QwtInterval::symmetrize (double value) const

Adjust the limit that is closer to value, so that value becomes the center of the interval.

Parameters

| value | Center |
|-------|--------|
| value | Ochto |

Returns

Interval with value as center

```
12.33.4.27 double QwtInterval::width ( ) const [inline]
```

Return the width of an interval.

The width of invalid intervals is 0.0, otherwise the result is maxValue() - minValue().

Returns

Interval width

See Also

isValid()

12.34 QwtIntervalSample Class Reference

A sample of the types (x1-x2, y) or (x, y1-y2)

```
#include <qwt_samples.h>
```

Public Member Functions

- QwtIntervalSample ()
- QwtIntervalSample (double, const QwtInterval &)

Constructor

• QwtIntervalSample (double value, double min, double max)

Constructor.

• bool operator== (const QwtIntervalSample &) const

Compare operator.

bool operator!= (const QwtIntervalSample &) const

Compare operator.

Public Attributes

• double value

Value.

QwtInterval interval

Interval.

12.34.1 Detailed Description

A sample of the types (x1-x2, y) or (x, y1-y2)

12.34.2 Constructor & Destructor Documentation

12.34.2.1 QwtIntervalSample::QwtIntervalSample() [inline]

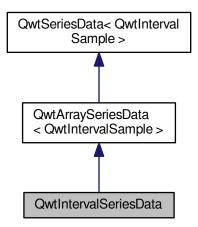
Constructor The value is set to 0.0, the interval is invalid

12.35 QwtIntervalSeriesData Class Reference

Interface for iterating over an array of intervals.

#include <qwt_series_data.h>

Inheritance diagram for QwtIntervalSeriesData:



Public Member Functions

- QwtIntervalSeriesData (const QVector< QwtIntervalSample > &=QVector< QwtIntervalSample >())
- · virtual QRectF boundingRect () const

Calculate the bounding rectangle.

Additional Inherited Members

12.35.1 Detailed Description

Interface for iterating over an array of intervals.

12.35.2 Constructor & Destructor Documentation

12.35.2.1 QwtIntervalSeriesData::QwtIntervalSeriesData (const QVector< QwtIntervalSample > & samples = QVector<QwtIntervalSample> ())

Constructor

Parameters

samples Samples

12.35.3 Member Function Documentation

12.35.3.1 QRectF QwtIntervalSeriesData::boundingRect() const [virtual]

Calculate the bounding rectangle.

The bounding rectangle is calculated once by iterating over all points and is stored for all following requests.

Returns

Bounding rectangle

Implements QwtSeriesData < QwtIntervalSample >.

12.36 QwtIntervalSymbol Class Reference

A drawing primitive for displaying an interval like an error bar.

```
#include <qwt_interval_symbol.h>
```

Public Types

 enum Style { NoSymbol = -1, Bar, Box, UserSymbol = 1000 } Symbol style.

Public Member Functions

- QwtIntervalSymbol (Style=NoSymbol)
- QwtIntervalSymbol (const QwtIntervalSymbol &)

Copy constructor.

virtual ~QwtIntervalSymbol ()

Destructor

QwtIntervalSymbol & operator= (const QwtIntervalSymbol &)

Assignment operator.

bool operator== (const QwtIntervalSymbol &) const

Compare two symbols.

• bool operator!= (const QwtIntervalSymbol &) const

Compare two symbols.

- · void setWidth (int)
- int width () const
- void setBrush (const QBrush &b)

Assign a brush.

- · const QBrush & brush () const
- void setPen (const QColor &, qreal width=0.0, Qt::PenStyle=Qt::SolidLine)
- void setPen (const QPen &)
- const QPen & pen () const
- void setStyle (Style)
- Style style () const
- virtual void draw (QPainter *, Qt::Orientation, const QPointF &from, const QPointF &to) const

12.36.1 Detailed Description

A drawing primitive for displaying an interval like an error bar.

See Also

QwtPlotIntervalCurve

12.36.2 Member Enumeration Documentation

12.36.2.1 enum QwtIntervalSymbol::Style

Symbol style.

Enumerator

NoSymbol No Style. The symbol cannot be drawn.

Bar The symbol displays a line with caps at the beginning/end. The size of the caps depends on the symbol width().

Box The symbol displays a plain rectangle using pen() and brush(). The size of the rectangle depends on the translated interval and the width(),

UserSymbol Styles >= UserSymbol are reserved for derived classes of QwtIntervalSymbol that overload draw() with additional application specific symbol types.

12.36.3 Constructor & Destructor Documentation

12.36.3.1 QwtIntervalSymbol::QwtIntervalSymbol (Style style = NoSymbol)

Constructor

Parameters

```
style Style of the symbol
```

See Also

```
setStyle(), style(), Style
```

12.36.4 Member Function Documentation

12.36.4.1 const QBrush & QwtIntervalSymbol::brush () const

Returns

Brush

See Also

setBrush()

12.36.4.2 void QwtIntervalSymbol::draw (QPainter * painter, Qt::Orientation orientation, const QPointF & from, const QPointF & to) const [virtual]

Draw a symbol depending on its style

Parameters

| painter | Painter |
|-------------|--|
| orientation | Orientation |
| from | Start point of the interval in target device coordinates |
| to | End point of the interval in target device coordinates |

See Also

setStyle()

12.36.4.3 const QPen & QwtIntervalSymbol::pen () const

Returns

Pen

See Also

setPen(), brush()

12.36.4.4 void QwtIntervalSymbol::setBrush (const QBrush & brush)

Assign a brush.

The brush is used for the Box style.

Parameters

| brush | Brush |
|-------|-------|
| | 2.00 |

See Also

brush()

12.36.4.5 void QwtIntervalSymbol::setPen (const QColor & color, qreal width = 0.0, Qt::PenStyle style = Qt::SolidLine)

Build and assign a pen

In Qt5 the default pen width is 1.0 (0.0 in Qt4) what makes it non cosmetic (see QPen::isCosmetic()). This method has been introduced to hide this incompatibility.

Parameters

| color | Pen color |
|-------|-----------|
| width | Pen width |
| style | Pen style |

See Also

pen(), brush()

12.36.4.6 void QwtIntervalSymbol::setPen (const QPen & pen)

Assign a pen

Parameters

pen Pen

See Also

pen(), setBrush()

12.36.4.7 void QwtIntervalSymbol::setStyle (Style style)

Specify the symbol style

Parameters

style Style

See Also

style(), Style

12.36.4.8 void QwtIntervalSymbol::setWidth (int width)

Specify the width of the symbol It is used depending on the style.

Parameters

width Width

See Also

width(), setStyle()

12.36.4.9 QwtIntervalSymbol::Style QwtIntervalSymbol::style () const

Returns

Current symbol style

See Also

setStyle()

12.36.4.10 int QwtIntervalSymbol::width () const

Returns

Width of the symbol.

See Also

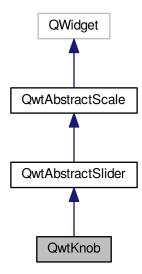
setWidth(), setStyle()

12.37 QwtKnob Class Reference

The Knob Widget.

#include <qwt_knob.h>

Inheritance diagram for QwtKnob:



Public Types

enum KnobStyle { Flat, Raised, Sunken, Styled }

Style of the knob surface.

• enum MarkerStyle {
 NoMarker = -1, Tick, Triangle, Dot,
 Nub, Notch }

Marker type.

Public Member Functions

QwtKnob (QWidget *parent=NULL)

Constructor.

virtual ~QwtKnob ()

Destructor.

void setAlignment (Qt::Alignment)

Set the alignment of the knob.

- Qt::Alignment alignment () const
- void setKnobWidth (int)

Change the knob's width.

• int knobWidth () const

Return the width of the knob.

void setNumTurns (int)

Set the number of turns.

- int numTurns () const
- void setTotalAngle (double angle)

Set the total angle by which the knob can be turned.

• double totalAngle () const

void setKnobStyle (KnobStyle)

Set the knob type.

- KnobStyle knobStyle () const
- void setBorderWidth (int bw)

Set the knob's border width.

• int borderWidth () const

Return the border width.

void setMarkerStyle (MarkerStyle)

Set the marker type of the knob.

- MarkerStyle markerStyle () const
- void setMarkerSize (int)

Set the size of the marker.

- int markerSize () const
- · virtual QSize sizeHint () const
- virtual QSize minimumSizeHint () const
- void setScaleDraw (QwtRoundScaleDraw *)
- const QwtRoundScaleDraw * scaleDraw () const
- QwtRoundScaleDraw * scaleDraw ()
- QRect knobRect () const

Protected Member Functions

- virtual void paintEvent (QPaintEvent *)
- virtual void changeEvent (QEvent *)
- virtual void drawKnob (QPainter *, const QRectF &) const

Draw the knob.

- virtual void drawFocusIndicator (QPainter *) const
- virtual void drawMarker (QPainter *, const QRectF &, double arc) const

Draw the marker at the knob's front.

virtual double scrolledTo (const QPoint &) const

Determine the value for a new position of the mouse.

virtual bool isScrollPosition (const QPoint &) const

Determine what to do when the user presses a mouse button.

Additional Inherited Members

12.37.1 Detailed Description

The Knob Widget.

The QwtKnob widget imitates look and behavior of a volume knob on a radio. It looks similar to QDial - not to QwtDial.

The value range of a knob might be divided into several turns.

The layout of the knob depends on the knobWidth().

- width > 0 The diameter of the knob is fixed and the knob is aligned according to the alignment() flags inside of the contentsRect().
- width <= 0 The knob is extended to the minimum of width/height of the contentsRect() and aligned in the other direction according to alignment().

Setting a fixed knobWidth() is helpful to align several knobs with different scale labels.

12.37.2 Member Enumeration Documentation

12.37.2.1 enum QwtKnob::KnobStyle

Style of the knob surface.

Depending on the KnobStyle the surface of the knob is filled from the brushes of the widget palette().

See Also

setKnobStyle(), knobStyle()

Enumerator

Flat Fill the knob with a brush from QPalette::Button.

Raised Build a gradient from QPalette::Midlight and QPalette::Button.

Sunken Build a gradient from QPalette::Midlight, QPalette::Button and QPalette::Midlight

Styled Build a radial gradient from QPalette::Button like it is used for QDial in various Qt styles.

12.37.2.2 enum QwtKnob::MarkerStyle

Marker type.

The marker indicates the current value on the knob The default setting is a Notch marker.

See Also

setMarkerStyle(), setMarkerSize()

Enumerator

NoMarker Don't paint any marker.

Tick Paint a single tick in QPalette::ButtonText color.

Triangle Paint a triangle in QPalette::ButtonText color.

Dot Paint a circle in QPalette::ButtonText color.

Nub Draw a raised ellipse with a gradient build from QPalette::Light and QPalette::Mid

Notch Draw a sunken ellipse with a gradient build from QPalette::Light and QPalette::Mid

12.37.3 Constructor & Destructor Documentation

12.37.3.1 QwtKnob::QwtKnob (QWidget * parent = NULL) [explicit]

Constructor.

Construct a knob with an angle of 270°. The style is QwtKnob::Raised and the marker style is QwtKnob::Notch. The width of the knob is set to 50 pixels.

Parameters

parent Parent widget

See Also

setTotalAngle()

12.37.4 Member Function Documentation

12.37.4.1 Qt::Alignment QwtKnob::alignment () const

Returns

Alignment of the knob inside of contentsRect()

See Also

setAlignment(), knobWidth(), knobRect()

12.37.4.2 void QwtKnob::changeEvent (QEvent * event) [protected], [virtual]

Handle QEvent::StyleChange and QEvent::FontChange;

Parameters

| event | Change event |
|-------|--------------|

12.37.4.3 void QwtKnob::drawFocusIndicator(QPainter * painter) const [protected], [virtual]

Draw the focus indicator

Parameters

| painter | Painter |
|---------|---------|
|---------|---------|

12.37.4.4 void QwtKnob::drawKnob (QPainter * painter, const QRectF & knobRect) const [protected], [virtual]

Draw the knob.

Parameters

| painter | painter |
|----------|--|
| knobRect | Bounding rectangle of the knob (without scale) |

12.37.4.5 void QwtKnob::drawMarker (QPainter * painter, const QRectF & rect, double angle) const [protected], [virtual]

Draw the marker at the knob's front.

Parameters

| painter | Painter |
|---------|--|
| rect | Bounding rectangle of the knob without scale |
| angle | Angle of the marker in degrees (clockwise, 0 at the 12 o'clock position) |

12.37.4.6 bool QwtKnob::isScrollPosition (const QPoint & pos) const [protected], [virtual]

Determine what to do when the user presses a mouse button.

Parameters

| pos Mouse position |
|--------------------|
|--------------------|

Return values

| True,when | pos is inside the circle of the knob. |
|-----------|---------------------------------------|

See Also

scrolledTo()

Implements QwtAbstractSlider.

```
12.37.4.7 QRect QwtKnob::knobRect ( ) const
Calculate the bounding rectangle of the knob without the scale
Returns
     Bounding rectangle of the knob
See Also
     knobWidth(), alignment(), QWidget::contentsRect()
12.37.4.8 QwtKnob::KnobStyle QwtKnob::knobStyle ( ) const
Returns
     Marker type of the knob
See Also
     setKnobStyle(), setBorderWidth()
12.37.4.9 int QwtKnob::markerSize ( ) const
Returns
     Marker size
See Also
     setMarkerSize()
12.37.4.10 QwtKnob::MarkerStyle QwtKnob::markerStyle ( ) const
Returns
     Marker type of the knob
See Also
     setMarkerStyle(), setMarkerSize()
12.37.4.11 QSize QwtKnob::minimumSizeHint() const [virtual]
Returns
     Minimum size hint
See Also
     sizeHint()
12.37.4.12 int QwtKnob::numTurns ( ) const
Returns
     Number of turns.
When the total angle is below 360° numTurns() is ceiled to 1.
See Also
     setNumTurns(), setTotalAngle(), totalAngle()
```

12.37.4.13 void QwtKnob::paintEvent (QPaintEvent * event) [protected], [virtual]

Repaint the knob

Parameters

event Paint event 12.37.4.14 const QwtRoundScaleDraw * QwtKnob::scaleDraw () const Returns the scale draw of the knob See Also setScaleDraw() 12.37.4.15 QwtRoundScaleDraw * QwtKnob::scaleDraw () Returns the scale draw of the knob See Also setScaleDraw() 12.37.4.16 double QwtKnob::scrolledTo(const QPoint & pos) const [protected], [virtual] Determine the value for a new position of the mouse. **Parameters** pos Mouse position Returns Value for the mouse position See Also isScrollPosition() Implements QwtAbstractSlider. 12.37.4.17 void QwtKnob::setAlignment (Qt::Alignment alignment) Set the alignment of the knob. Similar to a QLabel::alignment() the flags decide how to align the knob inside of contentsRect(). The default setting is Qt::AlignCenter **Parameters** alignment | Or'd alignment flags

See Also

alignment(), setKnobWidth(), knobRect()

12.37.4.18 void QwtKnob::setBorderWidth (int borderWidth)

Set the knob's border width.

Parameters

borderWidth new border width

12.37.4.19 void QwtKnob::setKnobStyle (KnobStyle knobStyle)

Set the knob type.

Parameters

knobStyle Knob type

See Also

knobStyle(), setBorderWidth()

12.37.4.20 void QwtKnob::setKnobWidth (int width)

Change the knob's width.

Setting a fixed value for the diameter of the knob is helpful for aligning several knobs in a row.

Parameters

width New width

See Also

knobWidth(), setAlignment()

Note

Modifies the sizePolicy()

12.37.4.21 void QwtKnob::setMarkerSize (int size)

Set the size of the marker.

When setting a size <= 0 the marker will automatically scaled to 40% of the radius of the knob.

See Also

markerSize(), markerStyle()

12.37.4.22 void QwtKnob::setMarkerStyle (MarkerStyle markerStyle)

Set the marker type of the knob.

Parameters

markerStyle Marker type

See Also

markerStyle(), setMarkerSize()

12.37.4.23 void QwtKnob::setNumTurns (int numTurns)

Set the number of turns.

When numTurns > 1 the knob can be turned several times around its axis

• otherwise the total angle is floored to 360°.

```
See Also
```

```
numTurns(), totalAngle(), setTotalAngle()
```

12.37.4.24 void QwtKnob::setScaleDraw (QwtRoundScaleDraw * scaleDraw)

Change the scale draw of the knob

For changing the labels of the scales, it is necessary to derive from QwtRoundScaleDraw and overload QwtRound-ScaleDraw::label().

See Also

scaleDraw()

12.37.4.25 void QwtKnob::setTotalAngle (double angle)

Set the total angle by which the knob can be turned.

Parameters

```
angle Angle in degrees.
```

The angle has to be between [10, 360] degrees. Angles above 360 (so that the knob can be turned several times around its axis) have to be set using setNumTurns().

The default angle is 270 degrees.

See Also

```
totalAngle(), setNumTurns()
```

```
12.37.4.26 QSize QwtKnob::sizeHint() const [virtual]
```

Returns

sizeHint()

12.37.4.27 double QwtKnob::totalAngle () const

Returns

the total angle

See Also

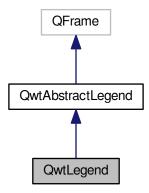
setTotalAngle(), setNumTurns(), numTurns()

12.38 QwtLegend Class Reference

The legend widget.

#include <qwt_legend.h>

Inheritance diagram for QwtLegend:



Public Slots

virtual void updateLegend (const QVariant &, const QList< QwtLegendData > &)
 Update the entries for an item.

Signals

- void clicked (const QVariant &itemInfo, int index)
- void checked (const QVariant &itemInfo, bool on, int index)

Public Member Functions

- QwtLegend (QWidget *parent=NULL)
- virtual ~QwtLegend ()

Destructor.

void setMaxColumns (uint numColums)

Set the maximum number of entries in a row.

- uint maxColumns () const
- void setDefaultItemMode (QwtLegendData::Mode)

Set the default mode for legend labels.

- QwtLegendData::Mode defaultItemMode () const
- QWidget * contentsWidget ()
- const QWidget * contentsWidget () const
- QWidget * legendWidget (const QVariant &) const

- QList< QWidget * > legendWidgets (const QVariant &) const
- QVariant itemInfo (const QWidget *) const
- virtual bool eventFilter (QObject *, QEvent *)
- · virtual QSize sizeHint () const

Return a size hint.

- · virtual int heightForWidth (int w) const
- QScrollBar * horizontalScrollBar () const
- QScrollBar * verticalScrollBar () const
- virtual void renderLegend (QPainter *, const QRectF &, bool fillBackground) const
- virtual void renderItem (QPainter *, const QWidget *, const QRectF &, bool fillBackground) const
- virtual bool is Empty () const
- virtual int scrollExtent (Qt::Orientation) const

Protected Slots

- void itemClicked ()
- void itemChecked (bool)

Protected Member Functions

virtual QWidget * createWidget (const QwtLegendData &) const

Create a widget to be inserted into the legend.

• virtual void updateWidget (QWidget *widget, const QwtLegendData &data)

Update the widget.

12.38.1 Detailed Description

The legend widget.

The QwtLegend widget is a tabular arrangement of legend items. Legend items might be any type of widget, but in general they will be a QwtLegendLabel.

See Also

QwtLegendLabel, QwtPlotItem, QwtPlot

12.38.2 Constructor & Destructor Documentation

```
12.38.2.1 QwtLegend::QwtLegend ( QWidget * parent = NULL ) [explicit]
```

Constructor

Parameters

| parent | Parent widget |
|--------|---------------|

12.38.3 Member Function Documentation

12.38.3.1 void QwtLegend::checked (const QVariant & itemInfo, bool on, int index) [signal]

A signal which is emitted when the user has clicked on a legend label, which is in QwtLegendData::Checkable mode

Parameters

| itemInfo | Info for the item of the selected legend label |
|----------|---|
| index | Index of the legend label in the list of widgets that are associated with the plot item |
| on | True when the legend label is checked |

Note

clicks are disabled as default

See Also

setDefaultItemMode(), defaultItemMode(), QwtPlot::itemToInfo()

12.38.3.2 void QwtLegend::clicked (const QVariant & itemInfo, int index) [signal]

A signal which is emitted when the user has clicked on a legend label, which is in QwtLegendData::Clickable mode.

Parameters

| itemInfo | Info for the item item of the selected legend item |
|----------|---|
| index | Index of the legend label in the list of widgets that are associated with the plot item |

Note

clicks are disabled as default

See Also

setDefaultItemMode(), defaultItemMode(), QwtPlot::itemToInfo()

12.38.3.3 QWidget * QwtLegend::contentsWidget ()

The contents widget is the only child of the viewport of the internal QScrollArea and the parent widget of all legend items.

Returns

Container widget of the legend items

12.38.3.4 const QWidget * QwtLegend::contentsWidget () const

The contents widget is the only child of the viewport of the internal QScrollArea and the parent widget of all legend items.

Returns

Container widget of the legend items

12.38.3.5 QWidget * QwtLegend::createWidget (const QwtLegendData & data) const [protected], [virtual]

Create a widget to be inserted into the legend.

The default implementation returns a QwtLegendLabel.

Parameters

| data | Attributes of the legend entry |
|------|--------------------------------|

Returns

Widget representing data on the legend

Note

updateWidget() will called soon after createWidget() with the same attributes.

12.38.3.6 QwtLegendData::Mode QwtLegend::defaultItemMode () const

Returns

Default item mode

See Also

setDefaultItemMode()

12.38.3.7 bool QwtLegend::eventFilter (QObject * object, QEvent * event) [virtual]

Handle QEvent::ChildRemoved andQEvent::LayoutRequest events for the contentsWidget().

Parameters

| object | Object to be filtered |
|--------|-----------------------|
| event | Event |

Returns

Forwarded to QwtAbstractLegend::eventFilter()

12.38.3.8 int QwtLegend::heightForWidth (int width) const [virtual]

Returns

The preferred height, for a width.

Parameters

| width | Width |
|-------|-------|

12.38.3.9 QScrollBar * QwtLegend::horizontalScrollBar () const

Returns

Horizontal scrollbar

See Also

verticalScrollBar()

12.38.3.10 bool QwtLegend::isEmpty()const [virtual]

Returns

True, when no item is inserted

Implements QwtAbstractLegend.

12.38.3.11 void QwtLegend::itemChecked (bool on) [protected], [slot]

Called internally when the legend has been checked Emits a checked() signal.

12.38.3.12 void QwtLegend::itemClicked() [protected], [slot]

Called internally when the legend has been clicked on. Emits a clicked() signal.

12.38.3.13 QVariant QwtLegend::itemInfo (const QWidget * widget) const

Find the item that is associated to a widget

Parameters

widget | Widget on the legend

Returns

Associated item info

See Also

legendWidget()

12.38.3.14 QWidget * QwtLegend::legendWidget (const QVariant & itemInfo) const

Returns

First widget in the list of widgets associated to an item

Parameters

itemInfo Info about an item

See Also

itemInfo(), QwtPlot::itemToInfo()

Note

Almost all types of items have only one widget

12.38.3.15 QList < QWidget * > QwtLegend::legendWidgets (const QVariant & itemInfo) const

Returns

List of widgets associated to a item

Parameters

itemInfo Info about an item

See Also

legendWidget(), itemInfo(), QwtPlot::itemToInfo()

12.38.3.16 uint QwtLegend::maxColumns () const

Returns

Maximum number of entries in a row

See Also

setMaxColumns(), QwtDynGridLayout::maxColumns()

12.38.3.17 void QwtLegend::renderItem (QPainter * painter, const QWidget * widget, const QRectF & rect, bool fillBackground) const [virtual]

Render a legend entry into a given rectangle.

Parameters

| painter | Painter |
|----------------|---|
| widget | Widget representing a legend entry |
| rect | Bounding rectangle |
| fillBackground | When true, fill rect with the widget background |

Note

When widget is not derived from QwtLegendLabel renderItem does nothing beside the background

12.38.3.18 void QwtLegend::renderLegend (QPainter * painter, const QRectF & rect, bool fillBackground) const [virtual]

Render the legend into a given rectangle.

Parameters

| painter | Painter |
|----------------|---|
| rect | Bounding rectangle |
| fillBackground | When true, fill rect with the widget background |

See Also

renderLegend() is used by QwtPlotRenderer - not by QwtLegend itself

Implements QwtAbstractLegend.

12.38.3.19 int QwtLegend::scrollExtent (Qt::Orientation orientation) const [virtual]

Return the extent, that is needed for the scrollbars

Parameters

| orienta | tation (| |
|---------|----------|--|

Returns

The width of the vertical scrollbar for Qt::Horizontal and v.v.

Reimplemented from QwtAbstractLegend.

12.38.3.20 void QwtLegend::setDefaultItemMode (QwtLegendData::Mode mode)

Set the default mode for legend labels.

Legend labels will be constructed according to the attributes in a QwtLegendData object. When it doesn't contain a value for the QwtLegendData::ModeRole the label will be initialized with the default mode of the legend.

Parameters

| mode | Default item mode |
|------|-------------------|
|------|-------------------|

See Also

itemMode(), QwtLegendData::value(), QwtPlotItem::legendData()

Note

Changing the mode doesn't have any effect on existing labels.

12.38.3.21 void QwtLegend::setMaxColumns (uint numColums)

Set the maximum number of entries in a row.

F.e when the maximum is set to 1 all items are aligned vertically. 0 means unlimited

Parameters

| numColums | Maximum number of entries in a row |
|-----------|------------------------------------|
|-----------|------------------------------------|

See Also

maxColumns(), QwtDynGridLayout::setMaxColumns()

12.38.3.22 void QwtLegend::updateLegend (const QVariant & *itemInfo*, const QList< QwtLegendData > & *data*) [virtual], [slot]

Update the entries for an item.

Parameters

| itemInfo | Info for an item |
|----------|--|
| data | List of legend entry attributes for the item |

12.38.3.23 void QwtLegend::updateWidget (QWidget * widget, const QwtLegendData & data) [protected], [virtual]

Update the widget.

Parameters

| widget | Usually a QwtLegendLabel |
|--------|----------------------------|
| data | Attributes to be displayed |

See Also

createWidget()

Note

When widget is no QwtLegendLabel updateWidget() does nothing.

12.38.3.24 QScrollBar * QwtLegend::verticalScrollBar () const

Returns

Vertical scrollbar

See Also

horizontalScrollBar()

12.39 QwtLegendData Class Reference

Attributes of an entry on a legend.

```
#include <qwt_legend_data.h>
```

Public Types

enum Mode { ReadOnly, Clickable, Checkable }

Mode defining how a legend entry interacts.

enum Role { ModeRole, TitleRole, IconRole, UserRole = 32 }

Identifier how to interprete a QVariant.

Public Member Functions

QwtLegendData ()

Constructor.

∼QwtLegendData ()

Destructor.

- void setValues (const QMap< int, QVariant > &)
- const QMap< int, QVariant > & values () const
- void setValue (int role, const QVariant &)
- QVariant value (int role) const
- · bool hasRole (int role) const
- bool isValid () const
- QwtGraphic icon () const
- QwtText title () const
- Mode mode () const

12.39.1 Detailed Description

Attributes of an entry on a legend.

QwtLegendData is an abstract container (like QAbstractModel) to exchange attributes, that are only known between to the plot item and the legend.

By overloading QwtPlotItem::legendData() any other set of attributes could be used, that can be handled by a modified (or completely different) implementation of a legend.

See Also

QwtLegend, QwtPlotLegendItem

Note

The stockchart example implements a legend as a tree with checkable items

12.39.2 Member Enumeration Documentation

12.39.2.1 enum QwtLegendData::Mode

Mode defining how a legend entry interacts.

Enumerator

ReadOnly The legend item is not interactive, like a label.

Clickable The legend item is clickable, like a push button.

Checkable The legend item is checkable, like a checkable button.

12.39.3 Member Function Documentation

12.39.3.1 bool QwtLegendData::hasRole (int role) const

Parameters

role Attribute role

Returns

True, when the internal map has an entry for role

12.39.3.2 QwtGraphic QwtLegendData::icon () const

Returns

Value of the IconRole attribute

12.39.3.3 bool QwtLegendData::isValid () const

Returns

True, when the internal map is empty

12.39.3.4 QwtLegendData::Mode QwtLegendData::mode () const

Returns

Value of the ModeRole attribute

12.39.3.5 void QwtLegendData::setValue (int role, const QVariant & data)

Set an attribute value

Parameters

| role | Attribute role |
|------|-----------------|
| data | Attribute value |

See Also

value()

12.39.3.6 void QwtLegendData::setValues (const QMap < int, QVariant > & map)

Set the legend attributes

QwtLegendData actually is a QMap<int, QVariant> with some convenience interfaces

Parameters

map Values

See Also

values()

12.39.3.7 QwtText QwtLegendData::title () const

Returns

Value of the TitleRole attribute

12.39.3.8 QVariant QwtLegendData::value (int role) const

Parameters

| role | Attribute role |
|------|----------------|
|------|----------------|

Returns

Attribute value for a specific role

12.39.3.9 const QMap < int, QVariant > & QwtLegendData::values () const

Returns

Legend attributes

See Also

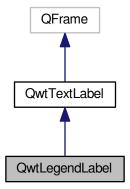
setValues()

12.40 QwtLegendLabel Class Reference

A widget representing something on a QwtLegend.

#include <qwt_legend_label.h>

Inheritance diagram for QwtLegendLabel:



Public Slots

• void setChecked (bool on)

Signals

• void clicked ()

Signal, when the legend item has been clicked.

· void pressed ()

Signal, when the legend item has been pressed.

• void released ()

Signal, when the legend item has been released.

void checked (bool)

Signal, when the legend item has been toggled.

Public Member Functions

- QwtLegendLabel (QWidget *parent=0)
- virtual ~QwtLegendLabel ()

Destructor.

- void setData (const QwtLegendData &)
- const QwtLegendData & data () const
- void setItemMode (QwtLegendData::Mode)
- QwtLegendData::Mode itemMode () const
- void setSpacing (int spacing)

Change the spacing between icon and text.

- int spacing () const
- virtual void setText (const QwtText &)
- void setlcon (const QPixmap &)
- QPixmap icon () const
- virtual QSize sizeHint () const

Return a size hint.

• bool isChecked () const

Return true, if the item is checked.

Protected Member Functions

void setDown (bool)

Set the item being down.

• bool isDown () const

Return true, if the item is down.

virtual void paintEvent (QPaintEvent *)

Paint event.

virtual void mousePressEvent (QMouseEvent *)

Handle mouse press events.

virtual void mouseReleaseEvent (QMouseEvent *)

Handle mouse release events.

virtual void keyPressEvent (QKeyEvent *)

Handle key press events.

virtual void keyReleaseEvent (QKeyEvent *)

Handle key release events.

12.40.1 Detailed Description

A widget representing something on a QwtLegend.

12.40.2 Constructor & Destructor Documentation

12.40.2.1 QwtLegendLabel::QwtLegendLabel(QWidget * parent = 0) [explicit]

Parameters

```
parent
                     Parent widget
12.40.3 Member Function Documentation
12.40.3.1 const QwtLegendData & QwtLegendLabel::data ( ) const
Returns
     Attributes of the label
See Also
     setData(), QwtPlotItem::legendData()
12.40.3.2 QPixmap QwtLegendLabel::icon ( ) const
Returns
     Pixmap representing a plot item
See Also
     setIcon()
12.40.3.3 QwtLegendData::Mode QwtLegendLabel::itemMode ( ) const
Returns
     Item mode
See Also
     setItemMode()
12.40.3.4 void QwtLegendLabel::setChecked ( bool on ) [slot]
Check/Uncheck a the item
Parameters
                     check/uncheck
                on
See Also
     setItemMode()
12.40.3.5 void QwtLegendLabel::setData ( const QwtLegendData & legendData )
Set the attributes of the legend label
Parameters
       legendData
                     Attributes of the label
See Also
     data()
12.40.3.6 void QwtLegendLabel::setIcon ( const QPixmap & icon )
Assign the icon
```

Parameters

icon Pixmap representing a plot item

See Also

icon(), QwtPlotItem::legendIcon()

12.40.3.7 void QwtLegendLabel::setItemMode (QwtLegendData::Mode mode)

Set the item mode The default is QwtLegendData::ReadOnly

Parameters

mode Item mode

See Also

itemMode()

12.40.3.8 void QwtLegendLabel::setSpacing (int spacing)

Change the spacing between icon and text.

Parameters

spacing Spacing

See Also

spacing(), QwtTextLabel::margin()

12.40.3.9 void QwtLegendLabel::setText (const QwtText & text) [virtual]

Set the text to the legend item

Parameters

text Text label

See Also

QwtTextLabel::text()

Reimplemented from QwtTextLabel.

12.40.3.10 int QwtLegendLabel::spacing () const

Returns

Spacing between icon and text

See Also

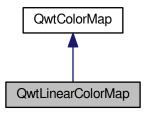
setSpacing(), QwtTextLabel::margin()

12.41 QwtLinearColorMap Class Reference

QwtLinearColorMap builds a color map from color stops.

#include <qwt_color_map.h>

Inheritance diagram for QwtLinearColorMap:



Public Types

• enum Mode { FixedColors, ScaledColors }

Public Member Functions

- QwtLinearColorMap (QwtColorMap::Format=QwtColorMap::RGB)
- QwtLinearColorMap (const QColor &from, const QColor &to, QwtColorMap::Format=QwtColorMap::RGB)
- virtual ~QwtLinearColorMap ()

Destructor.

void setMode (Mode)

Set the mode of the color map.

- Mode mode () const
- void setColorInterval (const QColor &color1, const QColor &color2)
- void addColorStop (double value, const QColor &)
- QVector< double > colorStops () const
- QColor color1 () const
- · QColor color2 () const
- virtual QRgb rgb (const QwtInterval &, double value) const
- · virtual unsigned char colorIndex (const QwtInterval &, double value) const

Map a value of a given interval into a color index.

12.41.1 Detailed Description

QwtLinearColorMap builds a color map from color stops.

A color stop is a color at a specific position. The valid range for the positions is [0.0, 1.0]. When mapping a value into a color it is translated into this interval according to mode().

12.41.2 Member Enumeration Documentation

12.41.2.1 enum QwtLinearColorMap::Mode

Mode of color map

See Also

setMode(), mode()

Enumerator

FixedColors Return the color from the next lower color stop.

ScaledColors Interpolating the colors of the adjacent stops.

12.41.3 Constructor & Destructor Documentation

12.41.3.1 QwtLinearColorMap::QwtLinearColorMap (QwtColorMap::Format format = QwtColorMap::RGB)

Build a color map with two stops at 0.0 and 1.0. The color at 0.0 is Qt::blue, at 1.0 it is Qt::yellow.

Parameters

| format | Preferred format of the color map |
|--------|-----------------------------------|

12.41.3.2 QwtLinearColorMap::QwtLinearColorMap (const QColor & color1, const QColor & color2, QwtColorMap::Format format = QwtColorMap::RGB)

Build a color map with two stops at 0.0 and 1.0.

Parameters

| color1 | Color used for the minimum value of the value interval |
|--------|--|
| color2 | Color used for the maximum value of the value interval |
| format | Preferred format for the color map |

12.41.4 Member Function Documentation

12.41.4.1 void QwtLinearColorMap::addColorStop (double value, const QColor & color)

Add a color stop

The value has to be in the range [0.0, 1.0]. F.e. a stop at position 17.0 for a range [10.0,20.0] must be passed as: (17.0 - 10.0) / (20.0 - 10.0)

Parameters

| value | Value between [0.0, 1.0] |
|-------|--------------------------|
| color | Color stop |

12.41.4.2 QColor QwtLinearColorMap::color1 () const

Returns

the first color of the color range

See Also

setColorInterval()

12.41.4.3 QColor QwtLinearColorMap::color2 () const

Returns

the second color of the color range

See Also

setColorInterval()

12.41.4.4 unsigned char QwtLinearColorMap::colorIndex (const QwtInterval & interval, double value) const [virtual]

Map a value of a given interval into a color index.

Parameters

| interval | Range for all values |
|----------|---------------------------------|
| value | Value to map into a color index |

Returns

Index, between 0 and 255

Implements QwtColorMap.

 ${\tt 12.41.4.5 \quad QVector} < {\tt double} > {\tt QwtLinearColorMap::colorStops} \ (\quad) \ {\tt const}$

Returns

Positions of color stops in increasing order

12.41.4.6 QwtLinearColorMap::Mode QwtLinearColorMap::mode () const

Returns

Mode of the color map

See Also

setMode()

12.41.4.7 QRgb QwtLinearColorMap::rgb (const QwtInterval & interval, double value) const [virtual]

Map a value of a given interval into a RGB value

Parameters

| interval | Range for all values |
|----------|-------------------------------|
| value | Value to map into a RGB value |

Returns

RGB value for value

Implements QwtColorMap.

12.41.4.8 void QwtLinearColorMap::setColorInterval (const QColor & color1, const QColor & color2)

Set the color range

Add stops at 0.0 and 1.0.

Parameters

| color1 | Color used for the minimum value of the value interval |
|--------|--|
| color2 | Color used for the maximum value of the value interval |

See Also

color1(), color2()

12.41.4.9 void QwtLinearColorMap::setMode (Mode mode)

Set the mode of the color map.

FixedColors means the color is calculated from the next lower color stop. ScaledColors means the color is calculated by interpolating the colors of the adjacent stops.

See Also

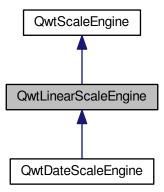
mode()

12.42 QwtLinearScaleEngine Class Reference

A scale engine for linear scales.

#include <qwt_scale_engine.h>

Inheritance diagram for QwtLinearScaleEngine:



Public Member Functions

- QwtLinearScaleEngine (uint base=10)
- virtual \sim QwtLinearScaleEngine ()

Destructor.

- virtual void autoScale (int maxSteps, double &x1, double &x2, double &stepSize) const
- virtual QwtScaleDiv divideScale (double x1, double x2, int numMajorSteps, int numMinorSteps, double step-Size=0.0) const

Calculate a scale division for an interval.

Protected Member Functions

QwtInterval align (const QwtInterval &, double stepSize) const

Align an interval to a step size.

• void buildTicks (const QwtInterval &, double stepSize, int maxMinSteps, QList< double > ticks[QwtScaleDiv:::NTickTypes]) const

Calculate ticks for an interval.

QList< double > buildMajorTicks (const QwtInterval &interval, double stepSize) const
 Calculate major ticks for an interval.

void buildMinorTicks (const QList< double > &majorTicks, int maxMinorSteps, double stepSize, QList< double > &minorTicks, QList< double > &mediumTicks) const

Calculate minor/medium ticks for major ticks.

Additional Inherited Members

12.42.1 Detailed Description

A scale engine for linear scales.

The step size will fit into the pattern $\{1,2,5\} \cdot 10^n$, where n is an integer.

12.42.2 Constructor & Destructor Documentation

12.42.2.1 QwtLinearScaleEngine::QwtLinearScaleEngine (uint base = 10)

Constructor

Parameters

| base | Base of the scale engine |
|------|--------------------------|

See Also

setBase()

12.42.3 Member Function Documentation

12.42.3.1 QwtInterval QwtLinearScaleEngine::align (const QwtInterval & interval, double stepSize) const [protected]

Align an interval to a step size.

The limits of an interval are aligned that both are integer multiples of the step size.

Parameters

| interval | Interval |
|----------|-----------|
| stepSize | Step size |

Returns

Aligned interval

12.42.3.2 void QwtLinearScaleEngine::autoScale (int *maxNumSteps*, double & *x1*, double & *x2*, double & *stepSize*) const [virtual]

Align and divide an interval

Parameters

| maxNumSteps | Max. number of steps |
|-------------|---------------------------------------|
| x1 | First limit of the interval (In/Out) |
| x2 | Second limit of the interval (In/Out) |
| stepSize | Step size (Out) |

See Also

setAttribute()

Implements QwtScaleEngine.

Reimplemented in QwtDateScaleEngine.

12.42.3.3 QList< double > QwtLinearScaleEngine::buildMajorTicks (const QwtInterval & interval, double stepSize) const [protected]

Calculate major ticks for an interval.

Parameters

| interval | Interval |
|----------|-----------|
| stepSize | Step size |

Returns

Calculated ticks

12.42.3.4 void QwtLinearScaleEngine::buildMinorTicks (const QList< double > & majorTicks, int maxMinorSteps, double stepSize, QList< double > & minorTicks, QList< double > & mediumTicks) const [protected]

Calculate minor/medium ticks for major ticks.

Parameters

| majorTicks | Major ticks |
|---------------|---|
| maxMinorSteps | Maximum number of minor steps |
| stepSize | Step size |
| minorTicks | Array to be filled with the calculated minor ticks |
| mediumTicks | Array to be filled with the calculated medium ticks |

12.42.3.5 void QwtLinearScaleEngine::buildTicks (const QwtInterval & interval, double stepSize, int maxMinorSteps, QList< double > ticks[QwtScaleDiv::NTickTypes]) const [protected]

Calculate ticks for an interval.

Parameters

| interval | Interval |
|---------------|---|
| stepSize | Step size |
| maxMinorSteps | Maximum number of minor steps |
| ticks | Arrays to be filled with the calculated ticks |

See Also

buildMajorTicks(), buildMinorTicks

12.42.3.6 QwtScaleDiv QwtLinearScaleEngine::divideScale (double x1, double x2, int maxMajorSteps, int maxMinorSteps, double stepSize = 0.0) const [virtual]

Calculate a scale division for an interval.

Parameters

| x1 | First interval limit |
|---------------|---|
| x2 | Second interval limit |
| maxMajorSteps | Maximum for the number of major steps |
| maxMinorSteps | Maximum number of minor steps |
| stepSize | Step size. If stepSize == 0, the engine calculates one. |

Returns

Calculated scale division

Implements QwtScaleEngine.

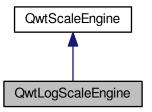
Reimplemented in QwtDateScaleEngine.

12.43 QwtLogScaleEngine Class Reference

A scale engine for logarithmic scales.

#include <qwt_scale_engine.h>

Inheritance diagram for QwtLogScaleEngine:



Public Member Functions

- QwtLogScaleEngine (uint base=10)
- virtual ~QwtLogScaleEngine ()

Destructor.

- virtual void autoScale (int maxSteps, double &x1, double &x2, double &stepSize) const
- virtual QwtScaleDiv divideScale (double x1, double x2, int numMajorSteps, int numMinorSteps, double step-Size=0.0) const

Calculate a scale division for an interval.

Protected Member Functions

QwtInterval align (const QwtInterval &, double stepSize) const

Align an interval to a step size.

• void buildTicks (const QwtInterval &, double stepSize, int maxMinSteps, QList< double > ticks[QwtScaleDiv:::NTickTypes]) const

Calculate ticks for an interval.

- QList < double > buildMajorTicks (const QwtInterval &interval, double stepSize) const
 Calculate major ticks for an interval.
- void buildMinorTicks (const QList< double > &majorTicks, int maxMinorSteps, double stepSize, QList< double > &minorTicks, QList< double > &mediumTicks) const

Calculate minor/medium ticks for major ticks.

Additional Inherited Members

12.43.1 Detailed Description

A scale engine for logarithmic scales.

The step size is measured in *decades* and the major step size will be adjusted to fit the pattern $\{1,2,3,5\} \cdot 10^n$, where n is a natural number including zero.

Warning

the step size as well as the margins are measured in decades.

12.43.2 Constructor & Destructor Documentation

12.43.2.1 QwtLogScaleEngine::QwtLogScaleEngine (uint base = 10)

Constructor

Parameters

| l | Description and a service |
|------|---------------------------|
| base | Base of the scale engine |
| Daoc | Base of the soule engine |
| | |

See Also

setBase()

12.43.3 Member Function Documentation

12.43.3.1 QwtInterval QwtLogScaleEngine::align (const QwtInterval & interval, double stepSize) const [protected]

Align an interval to a step size.

The limits of an interval are aligned that both are integer multiples of the step size.

Parameters

| interval | Interval |
|----------|-----------|
| stepSize | Step size |

Returns

Aligned interval

12.43.3.2 void QwtLogScaleEngine::autoScale (int *maxNumSteps*, double & *x1*, double & *x2*, double & *stepSize*) const [virtual]

Align and divide an interval

Parameters

| maxNumSte | s Max. number of steps |
|-----------|---------------------------------------|
|) | First limit of the interval (In/Out) |
|) | Second limit of the interval (In/Out) |
| stepSiz | e Step size (Out) |

See Also

QwtScaleEngine::setAttribute()

Implements QwtScaleEngine.

12.43.3.3 QList< double > QwtLogScaleEngine::buildMajorTicks (const QwtInterval & interval, double stepSize) const [protected]

Calculate major ticks for an interval.

Parameters

| interval | Interval |
|----------|-----------|
| stepSize | Step size |

Returns

Calculated ticks

12.43.3.4 void QwtLogScaleEngine::buildMinorTicks (const QList< double > & majorTicks, int maxMinorSteps, double stepSize, QList< double > & minorTicks, QList< double > & mediumTicks) const [protected]

Calculate minor/medium ticks for major ticks.

Parameters

| majorTicks | Major ticks |
|---------------|---|
| maxMinorSteps | Maximum number of minor steps |
| stepSize | Step size |
| minorTicks | Array to be filled with the calculated minor ticks |
| mediumTicks | Array to be filled with the calculated medium ticks |

12.43.3.5 void QwtLogScaleEngine::buildTicks (const QwtInterval & interval, double stepSize, int maxMinorSteps, QList< double > ticks[QwtScaleDiv::NTickTypes]) const [protected]

Calculate ticks for an interval.

Parameters

| interval | Interval |
|---------------|---|
| maxMinorSteps | Maximum number of minor steps |
| stepSize | Step size |
| ticks | Arrays to be filled with the calculated ticks |

See Also

buildMajorTicks(), buildMinorTicks

12.43.3.6 QwtScaleDiv QwtLogScaleEngine::divideScale (double x1, double x2, int maxMajorSteps, int maxMinorSteps, double stepSize = 0.0) const [virtual]

Calculate a scale division for an interval.

Parameters

| x1 | First interval limit |
|---------------|---|
| x2 | Second interval limit |
| maxMajorSteps | Maximum for the number of major steps |
| maxMinorSteps | Maximum number of minor steps |
| stepSize | Step size. If stepSize == 0, the engine calculates one. |

Returns

Calculated scale division

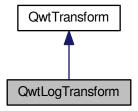
Implements QwtScaleEngine.

12.44 QwtLogTransform Class Reference

Logarithmic transformation.

#include <qwt_transform.h>

Inheritance diagram for QwtLogTransform:



Public Member Functions

• QwtLogTransform ()

Constructor.

virtual ~QwtLogTransform ()

Destructor.

- virtual double transform (double value) const
- virtual double invTransform (double value) const
- virtual double bounded (double value) const
- virtual QwtTransform * copy () const

Public Attributes

• QT_STATIC_CONST double LogMin = 1.0e-150

Smallest allowed value for logarithmic scales: 1.0e-150.

• QT_STATIC_CONST double LogMax = 1.0e150

Largest allowed value for logarithmic scales: 1.0e150.

12.44.1 Detailed Description

Logarithmic transformation.

QwtLogTransform modifies the values using log() and exp().

Note

In the calculations of QwtScaleMap the base of the log function has no effect on the mapping. So QwtLog-Transform can be used for log2(), log10() or any other logarithmic scale.

12.44.2 Member Function Documentation

12.44.2.1 double QwtLogTransform::bounded (double value) const [virtual]

Parameters

value Value to be bounded

Returns

qBound(LogMin, value, LogMax)

Reimplemented from QwtTransform.

12.44.2.2 QwtTransform * QwtLogTransform::copy() const [virtual]

Returns

Clone of the transformation

Implements QwtTransform.

12.44.2.3 double QwtLogTransform::invTransform (double value) const [virtual]

Parameters

value Value to be transformed

Returns

exp(value)

Implements QwtTransform.

12.44.2.4 double QwtLogTransform::transform (double value) const [virtual]

Parameters

value Value to be transformed

Returns

log(value)

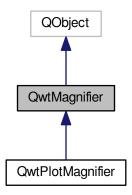
Implements QwtTransform.

12.45 QwtMagnifier Class Reference

QwtMagnifier provides zooming, by magnifying in steps.

#include <qwt_magnifier.h>

Inheritance diagram for QwtMagnifier:



Public Member Functions

- QwtMagnifier (QWidget *)
- virtual ~QwtMagnifier ()

Destructor.

- QWidget * parentWidget ()
- const QWidget * parentWidget () const
- · void setEnabled (bool)

En/disable the magnifier.

- · bool isEnabled () const
- void setMouseFactor (double)

Change the mouse factor.

- double mouseFactor () const
- void setMouseButton (Qt::MouseButton, Qt::KeyboardModifiers=Qt::NoModifier)
- void getMouseButton (Qt::MouseButton &, Qt::KeyboardModifiers &) const
- void setWheelFactor (double)

Change the wheel factor.

- · double wheelFactor () const
- void setWheelModifiers (Qt::KeyboardModifiers)
- Qt::KeyboardModifiers wheelModifiers () const
- void setKeyFactor (double)

Change the key factor.

- double keyFactor () const
- void setZoomInKey (int key, Qt::KeyboardModifiers=Qt::NoModifier)
- void getZoomInKey (int &key, Qt::KeyboardModifiers &) const

Retrieve the settings of the zoom in key.

- void setZoomOutKey (int key, Qt::KeyboardModifiers=Qt::NoModifier)
- void getZoomOutKey (int &key, Qt::KeyboardModifiers &) const

Retrieve the settings of the zoom out key.

virtual bool eventFilter (QObject *, QEvent *)

Event filter.

Protected Member Functions

- virtual void rescale (double factor)=0
- virtual void widgetMousePressEvent (QMouseEvent *)
- virtual void widgetMouseReleaseEvent (QMouseEvent *)
- virtual void widgetMouseMoveEvent (QMouseEvent *)
- virtual void widgetWheelEvent (QWheelEvent *)
- virtual void widgetKeyPressEvent (QKeyEvent *)
- virtual void widgetKeyReleaseEvent (QKeyEvent *)

12.45.1 Detailed Description

QwtMagnifier provides zooming, by magnifying in steps.

Using QwtMagnifier a plot can be zoomed in/out in steps using keys, the mouse wheel or moving a mouse button in vertical direction.

12.45.2 Constructor & Destructor Documentation

```
12.45.2.1 QwtMagnifier::QwtMagnifier ( QWidget * parent ) [explicit]
```

Constructor

Parameters

| parent | Widget to be magnified |
|--------|------------------------|

12.45.3 Member Function Documentation

```
12.45.3.1 bool QwtMagnifier::eventFilter ( QObject * object, QEvent * event ) [virtual]
```

Event filter.

When is Enabled() is true, the mouse events of the observed widget are filtered.

Parameters

| object | Object to be filtered |
|--------|-----------------------|
| event | Event |

Returns

Forwarded to QObject::eventFilter()

See Also

widgetMousePressEvent(), widgetMouseReleaseEvent(), widgetMouseMoveEvent(), widgetWheelEvent(), widgetKeyPressEvent() widgetKeyPressEvent()

12.45.3.2 void QwtMagnifier::getMouseButton (Qt::MouseButton & button, Qt::KeyboardModifiers & modifiers) const

See Also

setMouseButton()

12.45.3.3 void QwtMagnifier::getZoomlnKey (int & key, Qt::KeyboardModifiers & modifiers) const

Retrieve the settings of the zoom in key.

Parameters

| key | Key code, see Qt::Key |
|-----------|-----------------------|
| modifiers | Keyboard modifiers |

See Also

```
setZoomInKey()
```

12.45.3.4 void QwtMagnifier::getZoomOutKey (int & key, Qt::KeyboardModifiers & modifiers) const

Retrieve the settings of the zoom out key.

Parameters

| key | Key code, see Qt::Key |
|-----------|-----------------------|
| modifiers | Keyboard modifiers |

See Also

```
setZoomOutKey()
```

12.45.3.5 bool QwtMagnifier::isEnabled () const

Returns

true when enabled, false otherwise

See Also

```
setEnabled(), eventFilter()
```

12.45.3.6 double QwtMagnifier::keyFactor () const

Returns

Key factor

See Also

```
setKeyFactor()
```

12.45.3.7 double QwtMagnifier::mouseFactor () const

Returns

Mouse factor

See Also

setMouseFactor()

12.45.3.8 QWidget * QwtMagnifier::parentWidget ()

Returns

Parent widget, where the rescaling happens

```
12.45.3.9 const QWidget * QwtMagnifier::parentWidget ( ) const
```

Returns

Parent widget, where the rescaling happens

```
12.45.3.10 virtual void QwtMagnifier::rescale ( double factor ) [protected], [pure virtual]
```

Rescale the parent widget

Parameters

factor | Scale factor

Implemented in QwtPlotMagnifier.

12.45.3.11 void QwtMagnifier::setEnabled (bool on)

En/disable the magnifier.

When enabled is true an event filter is installed for the observed widget, otherwise the event filter is removed.

Parameters

on true or false

See Also

isEnabled(), eventFilter()

12.45.3.12 void QwtMagnifier::setKeyFactor (double factor)

Change the key factor.

The key factor defines the ratio between the current range on the parent widget and the zoomed range for each key press of the zoom in/out keys. The default value is 0.9.

Parameters

factor Key factor

See Also

keyFactor(), setZoomInKey(), setZoomOutKey(), setWheelFactor, setMouseFactor()

12.45.3.13 void QwtMagnifier::setMouseButton (Qt::MouseButton button, Qt::KeyboardModifiers modifiers = Qt::NoModifier)

Assign the mouse button, that is used for zooming in/out. The default value is Qt::RightButton.

Parameters

| button | Button |
|-----------|--------------------|
| modifiers | Keyboard modifiers |

See Also

getMouseButton()

12.45.3.14 void QwtMagnifier::setMouseFactor (double factor)

Change the mouse factor.

The mouse factor defines the ratio between the current range on the parent widget and the zoomed range for each vertical mouse movement. The default value is 0.95.

Parameters

factor Wheel factor

See Also

mouseFactor(), setMouseButton(), setWheelFactor(), setKeyFactor()

12.45.3.15 void QwtMagnifier::setWheelFactor (double factor)

Change the wheel factor.

The wheel factor defines the ratio between the current range on the parent widget and the zoomed range for each step of the wheel.

Use values > 1 for magnification (i.e. 2.0) and values < 1 for scaling down (i.e. 1/2.0 = 0.5). You can use this feature for inverting the direction of the wheel.

The default value is 0.9.

Parameters

| factor | Wheel factor |
|--------|--------------|

See Also

wheelFactor(), setWheelButtonState(), setMouseFactor(), setKeyFactor()

12.45.3.16 void QwtMagnifier::setWheelModifiers (Qt::KeyboardModifiers modifiers)

Assign keyboard modifiers for zooming in/out using the wheel. The default modifiers are Qt::NoModifiers.

Parameters

| modifiers | Keyboard modifiers |
|-----------|--------------------|

See Also

wheelModifiers()

12.45.3.17 void QwtMagnifier::setZoomlnKey (int key, Qt::KeyboardModifiers modifiers = Qt::NoModifier)

Assign the key, that is used for zooming in. The default combination is Qt::Key_Plus + Qt::NoModifier.

Parameters

| key | |
|-----------|--|
| modifiers | |

See Also

getZoomInKey(), setZoomOutKey()

12.45.3.18 void QwtMagnifier::setZoomOutKey (int key, Qt::KeyboardModifiers modifiers = Qt::NoModifier)

Assign the key, that is used for zooming out. The default combination is Qt::Key_Minus + Qt::NoModifier.

Parameters

| key | |
|-----------|--|
| modifiers | |

See Also

getZoomOutKey(), setZoomOutKey()

12.45.3.19 double QwtMagnifier::wheelFactor () const

Returns

Wheel factor

See Also

setWheelFactor()

12.45.3.20 Qt::KeyboardModifiers QwtMagnifier::wheelModifiers () const

Returns

Wheel modifiers

See Also

setWheelModifiers()

12.45.3.21 void QwtMagnifier::widgetKeyPressEvent (QKeyEvent * keyEvent) [protected], [virtual]

Handle a key press event for the observed widget.

Parameters

keyEvent Key event

See Also

eventFilter(), widgetKeyReleaseEvent()

12.45.3.22 void QwtMagnifier::widgetKeyReleaseEvent (QKeyEvent * keyEvent) [protected], [virtual]

Handle a key release event for the observed widget.

Parameters

keyEvent Key event

See Also

eventFilter(), widgetKeyReleaseEvent()

12.45.3.23 void QwtMagnifier::widgetMouseMoveEvent (QMouseEvent * mouseEvent) [protected], [virtual]

Handle a mouse move event for the observed widget.

Parameters

mouseEvent Mouse event

See Also

eventFilter(), widgetMousePressEvent(), widgetMouseReleaseEvent(),

12.45.3.24 void QwtMagnifier::widgetMousePressEvent (QMouseEvent * mouseEvent) [protected], [virtual]

Handle a mouse press event for the observed widget.

Parameters

| mouseEvent | Mouse event |
|------------|-------------|

See Also

eventFilter(), widgetMouseReleaseEvent(), widgetMouseMoveEvent()

12.45.3.25 void QwtMagnifier::widgetMouseReleaseEvent (QMouseEvent * mouseEvent) [protected], [virtual]

Handle a mouse release event for the observed widget.

Parameters

| mouseEvent | Mouse event |
|------------|-------------|
|------------|-------------|

See Also

eventFilter(), widgetMousePressEvent(), widgetMouseMoveEvent(),

12.45.3.26 void QwtMagnifier::widgetWheelEvent (QWheelEvent * wheelEvent) [protected], [virtual]

Handle a wheel event for the observed widget.

Parameters

| wheelEvent | Wheel event |
|------------|-------------|

See Also

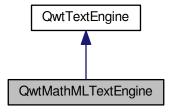
eventFilter()

12.46 QwtMathMLTextEngine Class Reference

Text Engine for the MathML renderer of the Qt solutions package.

#include <qwt_mathml_text_engine.h>

Inheritance diagram for QwtMathMLTextEngine:



Public Member Functions

QwtMathMLTextEngine ()

Constructor.

virtual ~QwtMathMLTextEngine ()

Destructor.

- virtual double heightForWidth (const QFont &font, int flags, const QString &text, double width) const
- virtual QSizeF textSize (const QFont &font, int flags, const QString &text) const
- · virtual void draw (QPainter *painter, const QRectF &rect, int flags, const QString &text) const
- virtual bool mightRender (const QString &) const
- virtual void textMargins (const QFont &, const QString &, double &left, double &right, double &top, double &bottom) const

Additional Inherited Members

12.46.1 Detailed Description

Text Engine for the MathML renderer of the Qt solutions package.

To enable MathML support the following code needs to be added to the application:

```
#include <qwt_mathml_text_engine.h>
QwtText::setTextEngine(QwtText::MathMLText, new QwtMathMLTextEngine());
```

See Also

QwtTextEngine, QwtText::setTextEngine

Warning

Unfortunately the MathML renderer doesn't support rotating of texts.

12.46.2 Member Function Documentation

12.46.2.1 void QwtMathMLTextEngine::draw (QPainter * painter, const QRectF & rect, int flags, const QString & text) const [virtual]

Draw the text in a clipping rectangle

Parameters

| painter | Painter |
|---------|--|
| rect | Clipping rectangle |
| flags | Bitwise OR of the flags like in for QPainter::drawText |
| text | Text to be rendered |

Implements QwtTextEngine.

12.46.2.2 double QwtMathMLTextEngine::heightForWidth (const QFont & font, int flags, const QString & text, double width) const [virtual]

Find the height for a given width

Parameters

| font | Font of the text |
|------|------------------|

| ſ | flags | Bitwise OR of the flags used like in QPainter::drawText |
|---|-------|---|
| Ī | text | Text to be rendered |
| Ī | width | Width |

Returns

Calculated height

Implements QwtTextEngine.

12.46.2.3 bool QwtMathMLTextEngine::mightRender (const QString & text) const [virtual]

Test if a string can be rendered by QwtMathMLTextEngine

Parameters

| text | Text to be tested |
|------|-------------------|

Returns

true, if text begins with "<math>".

Implements QwtTextEngine.

12.46.2.4 void QwtMathMLTextEngine::textMargins (const QFont & , const QString & , double & *left*, double & *right*, double & *top*, double & *bottom*) const [virtual]

Return margins around the texts

Parameters

| left | Return 0 |
|--------|----------|
| right | Return 0 |
| top | Return 0 |
| bottom | Return 0 |

Implements QwtTextEngine.

12.46.2.5 QSizeF QwtMathMLTextEngine::textSize (const QFont & font, int flags, const QString & text) const [virtual]

Returns the size, that is needed to render text

Parameters

| fon | Font of the text |
|-------|---|
| flags | Bitwise OR of the flags used like in QPainter::drawText |
| tex | Text to be rendered |

Returns

Caluclated size

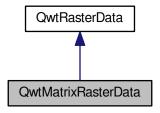
Implements QwtTextEngine.

12.47 QwtMatrixRasterData Class Reference

A class representing a matrix of values as raster data.

#include <qwt_matrix_raster_data.h>

Inheritance diagram for QwtMatrixRasterData:



Public Types

• enum ResampleMode { NearestNeighbour, BilinearInterpolation }

Resampling algorithm The default setting is NearestNeighbour;.

Public Member Functions

QwtMatrixRasterData ()

Constructor.

• virtual \sim QwtMatrixRasterData ()

Destructor.

void setResampleMode (ResampleMode mode)

Set the resampling algorithm.

- · ResampleMode resampleMode () const
- virtual void setInterval (Qt::Axis, const QwtInterval &)

Assign the bounding interval for an axis.

void setValueMatrix (const QVector< double > &values, int numColumns)

Assign a value matrix.

- const QVector< double > valueMatrix () const
- void setValue (int row, int col, double value)

Change a single value in the matrix.

- int numColumns () const
- int numRows () const
- virtual QRectF pixelHint (const QRectF &) const

Calculate the pixel hint.

• virtual double value (double x, double y) const

12.47.1 Detailed Description

A class representing a matrix of values as raster data.

QwtMatrixRasterData implements an interface for a matrix of equidistant values, that can be used by a QwtPlot-RasterItem. It implements a couple of resampling algorithms, to provide values for positions, that or not on the value matrix.

12.47.2 Member Enumeration Documentation

12.47.2.1 enum QwtMatrixRasterData::ResampleMode

Resampling algorithm The default setting is NearestNeighbour;.

Enumerator

NearestNeighbour Return the value from the matrix, that is nearest to the the requested position.

BilinearInterpolation Interpolate the value from the distances and values of the 4 surrounding values in the matrix,

12.47.3 Member Function Documentation

12.47.3.1 int QwtMatrixRasterData::numColumns () const

Returns

Number of columns of the value matrix

See Also

```
valueMatrix(), numRows(), setValueMatrix()
```

12.47.3.2 int QwtMatrixRasterData::numRows () const

Returns

Number of rows of the value matrix

See Also

```
valueMatrix(), numColumns(), setValueMatrix()
```

12.47.3.3 QRectF QwtMatrixRasterData::pixelHint(const QRectF & area) const [virtual]

Calculate the pixel hint.

pixelHint() returns the geometry of a pixel, that can be used to calculate the resolution and alignment of the plot item, that is representing the data.

- NearestNeighbour pixelHint() returns the surrounding pixel of the top left value in the matrix.
- BilinearInterpolation
 Returns an empty rectangle recommending to render in target device (f.e. screen) resolution.

Parameters

| area | Requested area, ignored |
|------|-------------------------|
|------|-------------------------|

Returns

Calculated hint

See Also

ResampleMode, setMatrix(), setInterval()

Reimplemented from QwtRasterData.

12.47.3.4 QwtMatrixRasterData::ResampleMode QwtMatrixRasterData::resampleMode () const

Returns

resampling algorithm

See Also

setResampleMode(), value()

12.47.3.5 void QwtMatrixRasterData::setInterval (Qt::Axis axis, const QwtInterval & interval) [virtual]

Assign the bounding interval for an axis.

Setting the bounding intervals for the X/Y axis is mandatory to define the positions for the values of the value matrix. The interval in Z direction defines the possible range for the values in the matrix, what is f.e used by QwtPlotSpectrogram to map values to colors. The Z-interval might be the bounding interval of the values in the matrix, but usually it isn't. (f.e a interval of 0.0-100.0 for values in percentage)

Parameters

| axis | X, Y or Z axis |
|----------|----------------|
| interval | Interval |

See Also

QwtRasterData::interval(), setValueMatrix()

Reimplemented from QwtRasterData.

12.47.3.6 void QwtMatrixRasterData::setResampleMode (ResampleMode mode)

Set the resampling algorithm.

Parameters

| mode | Resampling mode |
|------|-----------------|

See Also

resampleMode(), value()

12.47.3.7 void QwtMatrixRasterData::setValue (int row, int col, double value)

Change a single value in the matrix.

Parameters

| row | Row index |
|-------|--------------|
| col | Column index |
| value | New value |

See Also

value(), setValueMatrix()

12.47.3.8 void QwtMatrixRasterData::setValueMatrix (const QVector< double > & values, int numColumns)

Assign a value matrix.

The positions of the values are calculated by dividing the bounding rectangle of the X/Y intervals into equidistant rectangles (pixels). Each value corresponds to the center of a pixel.

Parameters

| values | Vector of values |
|------------|-------------------|
| numColumns | Number of columns |

See Also

valueMatrix(), numColumns(), numRows(), setInterval()()

12.47.3.9 double QwtMatrixRasterData::value (double x, double y) const [virtual]

Returns

the value at a raster position

Parameters

| Х | X value in plot coordinates |
|---|-----------------------------|
| У | Y value in plot coordinates |

See Also

ResampleMode

Implements QwtRasterData.

12.47.3.10 const QVector < double > QwtMatrixRasterData::valueMatrix () const

Returns

Value matrix

See Also

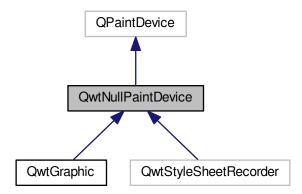
setValueMatrix(), numColumns(), numRows(), setInterval()

12.48 QwtNullPaintDevice Class Reference

A null paint device doing nothing.

#include <qwt_null_paintdevice.h>

Inheritance diagram for QwtNullPaintDevice:



Public Types

enum Mode { NormalMode, PolygonPathMode, PathMode }

Render mode.

Public Member Functions

QwtNullPaintDevice ()

Constructor.

virtual ~QwtNullPaintDevice ()

Destructor.

- void setMode (Mode)
- Mode mode () const
- virtual QPaintEngine * paintEngine () const

See QPaintDevice::paintEngine()

- virtual int metric (PaintDeviceMetric metric) const
- virtual void drawRects (const QRect *, int)

See QPaintEngine::drawRects()

virtual void drawRects (const QRectF *, int)

See QPaintEngine::drawRects()

virtual void drawLines (const QLine *, int)

See QPaintEngine::drawLines()

virtual void drawLines (const QLineF *, int)

See QPaintEngine::drawLines()

virtual void drawEllipse (const QRectF &)

See QPaintEngine::drawEllipse()

virtual void drawEllipse (const QRect &)

See QPaintEngine::drawEllipse()

• virtual void drawPath (const QPainterPath &)

See QPaintEngine::drawPath()

virtual void drawPoints (const QPointF *, int)

See QPaintEngine::drawPoints()

virtual void drawPoints (const QPoint *, int)

See QPaintEngine::drawPoints()

virtual void drawPolygon (const QPointF *, int, QPaintEngine::PolygonDrawMode)

See QPaintEngine::drawPolygon()

virtual void drawPolygon (const QPoint *, int, QPaintEngine::PolygonDrawMode)

See QPaintEngine::drawPolygon()

virtual void drawPixmap (const QRectF &, const QPixmap &, const QRectF &)

See QPaintEngine::drawPixmap()

virtual void drawTextItem (const QPointF &, const QTextItem &)

See QPaintEngine::drawTextItem()

virtual void drawTiledPixmap (const QRectF &, const QPixmap &, const QPointF &s)

See QPaintEngine::drawTiledPixmap()

virtual void drawlmage (const QRectF &, const QImage &, const QRectF &, Qt::ImageConversionFlags)

See QPaintEngine::drawImage()

virtual void updateState (const QPaintEngineState &state)

See QPaintEngine::updateState()

Protected Member Functions

• virtual QSize sizeMetrics () const =0

12.48.1 Detailed Description

A null paint device doing nothing.

Sometimes important layout/rendering geometries are not available or changeable from the public Qt class interface. (f.e hidden in the style implementation).

QwtNullPaintDevice can be used to manipulate or filter out this information by analyzing the stream of paint primitives.

F.e. QwtNullPaintDevice is used by QwtPlotCanvas to identify styled backgrounds with rounded corners.

12.48.2 Member Enumeration Documentation

12.48.2.1 enum QwtNullPaintDevice::Mode

Render mode.

See Also

setMode(), mode()

Enumerator

NormalMode All vector graphic primitives are painted by the corresponding draw methods

PolygonPathMode Vector graphic primitives (beside polygons) are mapped to a QPainterPath and are painted by drawPath. In PathMode mode only a few draw methods are called:

- drawPath()
- drawPixmap()
- drawlmage()
- drawPolygon()

PathMode Vector graphic primitives are mapped to a QPainterPath and are painted by drawPath. In PathMode mode only a few draw methods are called:

- drawPath()
- drawPixmap()
- drawlmage()

12.48.3 Member Function Documentation

12.48.3.1 int QwtNullPaintDevice::metric (PaintDeviceMetric deviceMetric) const [virtual]

See QPaintDevice::metric()

Parameters

| deviceMetric | Type of metric |
|--------------|----------------|

Returns

Metric information for the given paint device metric.

See Also

sizeMetrics()

12.48.3.2 QwtNullPaintDevice::Mode QwtNullPaintDevice::mode () const

Returns

Render mode

See Also

setMode()

12.48.3.3 void QwtNullPaintDevice::setMode (Mode mode)

Set the render mode

Parameters

mode New mode

See Also

mode()

12.48.3.4 virtual QSize QwtNullPaintDevice::sizeMetrics()const [protected], [pure virtual]

Returns

Size needed to implement metric()

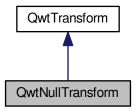
Implemented in QwtGraphic.

12.49 QwtNullTransform Class Reference

Null transformation.

#include <qwt_transform.h>

Inheritance diagram for QwtNullTransform:



Public Member Functions

• QwtNullTransform ()

Constructor.

virtual ~QwtNullTransform ()

Destructor.

- · virtual double transform (double value) const
- · virtual double invTransform (double value) const
- virtual QwtTransform * copy () const

12.49.1 Detailed Description

Null transformation.

QwtNullTransform returns the values unmodified.

12.49.2 Member Function Documentation

12.49.2.1 QwtTransform * QwtNullTransform::copy() const [virtual]

Returns

Clone of the transformation

Implements QwtTransform.

12.49.2.2 double QwtNullTransform::invTransform (double value) const [virtual]

Parameters

value Value to be transformed

Returns

value unmodified

Implements QwtTransform.

12.49.2.3 double QwtNullTransform::transform (double value) const [virtual]

Parameters

value Value to be transformed

Returns

value unmodified

Implements QwtTransform.

12.50 QwtOHLCSample Class Reference

Open-High-Low-Close sample used in financial charts.

#include <qwt_samples.h>

Public Member Functions

- QwtOHLCSample (double time=0.0, double open=0.0, double high=0.0, double low=0.0, double close=0.0)
- QwtInterval boundingInterval () const

Calculate the bounding interval of the OHLC values.

• bool isValid () const

Check if a sample is valid.

Public Attributes

- · double time
- · double open

Opening price.

· double high

Highest price.

· double low

Lowest price.

· double close

Closing price.

12.50.1 Detailed Description

Open-High-Low-Close sample used in financial charts.

In financial charts the movement of a price in a time interval is often represented by the opening/closing prices and the lowest/highest prices in this interval.

See Also

QwtTradingChartData

12.50.2 Constructor & Destructor Documentation

```
12.50.2.1 QwtOHLCSample::QwtOHLCSample ( double t = 0.0, double o = 0.0, double h = 0.0, double l = 0.0
```

Constructor

Parameters

| t | Time value |
|---|-------------|
| 0 | Open value |
| h | High value |
| 1 | Low value |
| С | Close value |

12.50.3 Member Function Documentation

12.50.3.1 QwtInterval QwtOHLCSample::boundingInterval () const [inline]

Calculate the bounding interval of the OHLC values.

For valid samples the limits of this interval are always low/high.

Returns

Bounding interval

See Also

isValid()

12.50.3.2 bool QwtOHLCSample::isValid () const [inline]

Check if a sample is valid.

A sample is valid, when all of the following checks are true:

- low <= high
- low <= open <= high
- low <= close <= high

Returns

True, when the sample is valid

12.50.4 Member Data Documentation

12.50.4.1 double QwtOHLCSample::time

Time of the sample, usually a number representing a specific interval - like a day.

12.51 QwtPainter Class Reference

A collection of QPainter workarounds.

```
#include <qwt_painter.h>
```

Static Public Member Functions

static void setPolylineSplitting (bool)

En/Disable line splitting for the raster paint engine.

- static bool polylineSplitting ()
- static void setRoundingAlignment (bool)
- static bool roundingAlignment ()
- static bool roundingAlignment (QPainter *)
- static void drawText (QPainter *, double x, double y, const QString &)

Wrapper for QPainter::drawText()

static void drawText (QPainter *, const QPointF &, const QString &)

Wrapper for QPainter::drawText()

• static void drawText (QPainter *, double x, double y, double w, double h, int flags, const QString &)

Wrapper for QPainter::drawText()

static void drawText (QPainter *, const QRectF &, int flags, const QString &)

Wrapper for QPainter::drawText()

- static void drawSimpleRichText (QPainter *, const QRectF &, int flags, const QTextDocument &)
- static void drawRect (QPainter *, double x, double y, double w, double h)

Wrapper for QPainter::drawRect()

static void drawRect (QPainter *, const QRectF &rect)

Wrapper for QPainter::drawRect()

• static void fillRect (QPainter *, const QRectF &, const QBrush &)

Wrapper for QPainter::fillRect()

static void drawEllipse (QPainter *, const QRectF &)

Wrapper for QPainter::drawEllipse()

static void drawPie (QPainter *, const QRectF &r, int a, int alen)

Wrapper for QPainter::drawPie() static void drawLine (QPainter *, double x1, double y1, double x2, double y2) Wrapper for QPainter::drawLine() static void drawLine (QPainter *, const QPointF &p1, const QPointF &p2) Wrapper for QPainter::drawLine() static void drawLine (QPainter *, const QLineF &) Wrapper for QPainter::drawLine() static void drawPolygon (QPainter *, const QPolygonF &) Wrapper for QPainter::drawPolygon() static void drawPolyline (QPainter *, const QPolygonF &) Wrapper for QPainter::drawPolyline() static void drawPolyline (QPainter *, const QPointF *, int pointCount) Wrapper for QPainter::drawPolyline() static void drawPolygon (QPainter *, const QPolygon &) Wrapper for QPainter::drawPolygon() static void drawPolyline (QPainter *, const QPolygon &) Wrapper for QPainter::drawPolyline() static void drawPolyline (QPainter *, const QPoint *, int pointCount) Wrapper for QPainter::drawPolyline() static void drawPoint (QPainter *, const QPoint &) Wrapper for QPainter::drawPoint() static void drawPoints (QPainter *, const QPolygon &) Wrapper for QPainter::drawPoints() static void drawPoints (QPainter *, const QPoint *, int pointCount) Wrapper for QPainter::drawPoints() static void drawPoint (QPainter *, double x, double y) Wrapper for QPainter::drawPoint() static void drawPoint (QPainter *, const QPointF &) Wrapper for QPainter::drawPoint() static void drawPoints (QPainter *, const QPolygonF &) Wrapper for QPainter::drawPoints() static void drawPoints (QPainter *, const QPointF *, int pointCount) Wrapper for QPainter::drawPoints() static void drawPath (QPainter *, const QPainterPath &) Wrapper for QPainter::drawPath() static void drawlmage (QPainter *, const QRectF &, const QImage &) Wrapper for QPainter::drawImage() static void drawPixmap (QPainter *, const QRectF &, const QPixmap &) Wrapper for QPainter::drawPixmap() static void drawRoundFrame (QPainter *, const QRectF &, const QPalette &, int lineWidth, int frameStyle) • static void drawRoundedFrame (QPainter *, const QRectF &, double xRadius, double yRadius, const QPalette &, int lineWidth, int frameStyle) • static void drawFrame (QPainter *, const QRectF &rect, const QPalette &palette, QPalette::ColorRole foregroundRole, int lineWidth, int midLineWidth, int frameStyle) static void drawFocusRect (QPainter *, const QWidget *) Draw a focus rectangle on a widget using its style. static void drawFocusRect (QPainter *, const QWidget *, const QRect &) Draw a focus rectangle on a widget using its style. • static void drawColorBar (QPainter *painter, const QwtColorMap &, const QwtInterval &, const QwtScaleMap &, Qt::Orientation, const QRectF &) static bool isAligning (QPainter *painter)

static QPixmap backingStore (QWidget *, const QSize &)

• static void fillPixmap (const QWidget *, QPixmap &, const QPoint &offset=QPoint())

static void drawBackgound (QPainter *painter, const QRectF &rect, const QWidget *widget)

• static bool isX11GraphicsSystem ()

12.51.1 Detailed Description

A collection of QPainter workarounds.

12.51.2 Member Function Documentation

12.51.2.1 QPixmap QwtPainter::backingStore (QWidget * widget, const QSize & size) [static]

Returns

A pixmap that can be used as backing store

Parameters

| widg | get | Widget, for which the backinstore is intended |
|------|-----|---|
| 31 | ze | Size of the pixmap |

12.51.2.2 void QwtPainter::drawBackgound (QPainter * painter, const QRectF & rect, const QWidget * widget) [static]

Fill rect with the background of a widget

Parameters

| painter | Painter |
|---------|------------------------|
| rect | Rectangle to be filled |
| widget | Widget |

See Also

QStyle::PE_Widget, QWidget::backgroundRole()

12.51.2.3 void QwtPainter::drawColorBar (QPainter * painter, const QwtColorMap & colorMap, const QwtInterval & interval, const QwtScaleMap & scaleMap, Qt::Orientation orientation, const QRectF & rect) [static]

Draw a color bar into a rectangle

Parameters

| painter | Painter |
|-------------|------------------|
| colorMap | Color map |
| interval | Value range |
| scaleMap | Scale map |
| orientation | Orientation |
| rect | Traget rectangle |

12.51.2.4 void QwtPainter::drawFrame (QPainter * painter, const QRectF & rect, const QPalette & palette, QPalette::ColorRole foregroundRole, int frameWidth, int midLineWidth, int frameStyle) [static]

Draw a rectangular frame

Parameters

| painter | Painter |
|---------|-----------------|
| rect | Frame rectangle |

| palette | Palette |
|----------------|---|
| foregroundRole | Foreground role used for QFrame::Plain |
| frameWidth | Frame width |
| midLineWidth | Used for QFrame::Box |
| frameStyle | bitwise OR'ed value of QFrame::Shape and QFrame::Shadow |

12.51.2.5 void QwtPainter::drawRoundedFrame (QPainter * painter, const QRectF & rect, double xRadius, double yRadius, const QPalette & palette, int lineWidth, int frameStyle) [static]

Draw a rectangular frame with rounded borders

Parameters

| painter | Painter |
|------------|--|
| rect | Frame rectangle |
| xRadius | x-radius of the ellipses defining the corners |
| yRadius | y-radius of the ellipses defining the corners |
| palette | QPalette::WindowText is used for plain borders QPalette::Dark and QPalette::Light for raised |
| | or sunken borders |
| lineWidth | Line width |
| frameStyle | bitwise OR´ed value of QFrame::Shape and QFrame::Shadow |

12.51.2.6 void QwtPainter::drawRoundFrame (QPainter * painter, const QRectF & rect, const QPalette & palette, int lineWidth, int frameStyle) [static]

Draw a round frame

Parameters

| painter | Painter |
|------------|--|
| rect | Frame rectangle |
| palette | QPalette::WindowText is used for plain borders QPalette::Dark and QPalette::Light for raised |
| | or sunken borders |
| lineWidth | Line width |
| frameStyle | bitwise OR´ed value of QFrame::Shape and QFrame::Shadow |

12.51.2.7 void QwtPainter::drawSimpleRichText (QPainter * painter, const QRectF & rect, int flags, const QTextDocument & text) [static]

Draw a text document into a rectangle

Parameters

| painter | Painter |
|---------|---|
| rect | Traget rectangle |
| flags | Alignments/Text flags, see QPainter::drawText() |
| text | Text document |

12.51.2.8 void QwtPainter::fillPixmap (const QWidget * widget, QPixmap & pixmap, const QPoint & offset = QPoint ()) [static]

Fill a pixmap with the content of a widget

In Qt >= 5.0 QPixmap::fill() is a nop, in Qt 4.x it is buggy for backgrounds with gradients. Thus fillPixmap() offers an alternative implementation.

Parameters

| widget | Widget |
|--------|---------------------|
| pixmap | Pixmap to be filled |
| offset | Offset |

See Also

QPixmap::fill()

12.51.2.9 bool QwtPainter::isAligning (QPainter * painter) [static]

Check if the painter is using a paint engine, that aligns coordinates to integers. Today these are all paint engines beside QPaintEngine::Pdf and QPaintEngine::SVG.

If we have an integer based paint engine it is also checked if the painter has a transformation matrix, that rotates or scales.

Parameters

| painter | Painter |
|---------|---------|
|---------|---------|

Returns

true, when the painter is aligning

See Also

setRoundingAlignment()

12.51.2.10 bool QwtPainter::isX11GraphicsSystem() [static]

Check is the application is running with the X11 graphics system that has some special capabilities that can be used for incremental painting to a widget.

Returns

True, when the graphics system is X11

12.51.2.11 bool QwtPainter::polylineSplitting() [inline], [static]

Returns

True, when line splitting for the raster paint engine is enabled.

See Also

setPolylineSplitting()

12.51.2.12 bool QwtPainter::roundingAlignment() [inline], [static]

Check whether coordinates should be rounded, before they are painted to a paint engine that rounds to integer values. For other paint engines (PDF, SVG), this flag has no effect.

Returns

True, when rounding is enabled

See Also

setRoundingAlignment(), isAligning()

```
12.51.2.13 bool QwtPainter::roundingAlignment( QPainter * painter) [inline], [static]
```

Returns

roundingAlignment() && isAligning(painter);

Parameters

| nainter | Painter |
|----------|---------|
| pairitei | Painter |

12.51.2.14 void QwtPainter::setPolylineSplitting (bool enable) [static]

En/Disable line splitting for the raster paint engine.

In some Qt versions the raster paint engine paints polylines of many points much faster when they are split in smaller chunks: f.e all supported Qt versions \geq = Qt 5.0 when drawing an antialiased polyline with a pen width \geq =2.

The default setting is true.

See Also

polylineSplitting()

```
12.51.2.15 void QwtPainter::setRoundingAlignment (bool enable) [static]
```

Enable whether coordinates should be rounded, before they are painted to a paint engine that floors to integer values. For other paint engines this (PDF, SVG), this flag has no effect. QwtPainter stores this flag only, the rounding itself is done in the painting code (f.e the plot items).

The default setting is true.

See Also

roundingAlignment(), isAligning()

12.52 QwtPainterCommand Class Reference

```
#include <qwt_painter_command.h>
```

Classes

struct ImageData

Attributes how to paint a QImage.

struct PixmapData

Attributes how to paint a QPixmap.

• struct StateData

Attributes of a state change.

Public Types

```
    enum Type {
        Invalid = -1, Path, Pixmap, Image,
        State }
```

Type of the paint command.

Public Member Functions

QwtPainterCommand ()

Construct an invalid command.

- QwtPainterCommand (const QwtPainterCommand &)
- QwtPainterCommand (const QPainterPath &)

Copy constructor.

- QwtPainterCommand (const QRectF &rect, const QPixmap &, const QRectF &subRect)
- QwtPainterCommand (const QRectF &rect, const QImage &, const QRectF &subRect, Qt::ImageConversion-Flags)
- QwtPainterCommand (const QPaintEngineState &)
- ∼QwtPainterCommand ()

Destructor.

- QwtPainterCommand & operator= (const QwtPainterCommand &)
- Type type () const
- QPainterPath * path ()
- const QPainterPath * path () const
- PixmapData * pixmapData ()
- const PixmapData * pixmapData () const
- ImageData * imageData ()
- const ImageData * imageData () const
- StateData * stateData ()
- const StateData * stateData () const

12.52.1 Detailed Description

QwtPainterCommand represents the attributes of a paint operation how it is used between QPainter and QPaint-Device

It is used by QwtGraphic to record and replay paint operations

See Also

QwtGraphic::commands()

12.52.2 Member Enumeration Documentation

12.52.2.1 enum QwtPainterCommand::Type

Type of the paint command.

Enumerator

Invalid Invalid command.

Path Draw a QPainterPath.

Pixmap Draw a QPixmap.

Image Draw a Qlmage.

State QPainter state change.

12.52.3 Constructor & Destructor Documentation

12.52.3.1 QwtPainterCommand::QwtPainterCommand (const QwtPainterCommand & other)

Copy constructor

Parameters

| other | Command to be copied |
|-------|----------------------|
|-------|----------------------|

12.52.3.2 QwtPainterCommand::QwtPainterCommand (const QRectF & rect, const QPixmap & pixmap, const QRectF & subRect)

Constructor for Pixmap paint operation

Parameters

| rect | Target rectangle |
|---------|-----------------------------|
| pixmap | Pixmap |
| subRect | Rectangle inside the pixmap |

See Also

QPainter::drawPixmap()

12.52.3.3 QwtPainterCommand::QwtPainterCommand (const QRectF & rect, const QImage & image, const QRectF & subRect, Qt::ImageConversionFlags flags)

Constructor for Image paint operation

Parameters

| rect | Target rectangle |
|---------|----------------------------|
| image | Image |
| subRect | Rectangle inside the image |
| flags | Conversion flags |

See Also

QPainter::drawImage()

12.52.3.4 QwtPainterCommand::QwtPainterCommand (const QPaintEngineState & state)

Constructor for State paint operation

Parameters

| state | Paint engine state |
|-------|--------------------|

12.52.4 Member Function Documentation

12.52.4.1 QwtPainterCommand::ImageData * QwtPainterCommand::imageData ()

Returns

Attributes how to paint a QImage

12.52.4.2 const QwtPainterCommand::ImageData * QwtPainterCommand::imageData () const [inline]

Returns

Attributes how to paint a QImage

12.52.4.3 QwtPainterCommand & QwtPainterCommand & other)

Assignment operator

```
Parameters
```

```
other
                     Command to be copied
Returns
     Modified command
12.52.4.4 QPainterPath * QwtPainterCommand::path ( )
Returns
     Painter path to be painted
12.52.4.5 const QPainterPath * QwtPainterCommand::path ( ) const [inline]
Returns
     Painter path to be painted
12.52.4.6 QwtPainterCommand::PixmapData * QwtPainterCommand::pixmapData ( )
Returns
     Attributes how to paint a QPixmap
12.52.4.7 const QwtPainterCommand::PixmapData * QwtPainterCommand::pixmapData ( ) const [inline]
Returns
     Attributes how to paint a QPixmap
12.52.4.8 QwtPainterCommand::StateData * QwtPainterCommand::stateData ( )
Returns
```

Attributes of a state change

12.52.4.9 const QwtPainterCommand::StateData * QwtPainterCommand::stateData () const [inline]

Attributes of a state change

12.52.4.10 QwtPainterCommand::Type QwtPainterCommand::type()const [inline]

Returns

Returns

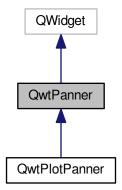
Type of the command

12.53 QwtPanner Class Reference

QwtPanner provides panning of a widget.

#include <qwt_panner.h>

Inheritance diagram for QwtPanner:



Signals

- void panned (int dx, int dy)
- void moved (int dx, int dy)

Public Member Functions

- QwtPanner (QWidget *parent)
- virtual ~QwtPanner ()

Destructor.

void setEnabled (bool)

En/disable the panner.

- bool isEnabled () const
- void setMouseButton (Qt::MouseButton, Qt::KeyboardModifiers=Qt::NoModifier)
- void getMouseButton (Qt::MouseButton &button, Qt::KeyboardModifiers &) const

Get mouse button and modifiers used for panning.

- void setAbortKey (int key, Qt::KeyboardModifiers=Qt::NoModifier)
- void getAbortKey (int &key, Qt::KeyboardModifiers &) const

Get the abort key and modifiers.

- void setCursor (const QCursor &)
- const QCursor cursor () const
- void setOrientations (Qt::Orientations)
- Qt::Orientations orientations () const

Return the orientation, where paning is enabled.

- bool isOrientationEnabled (Qt::Orientation) const
- virtual bool eventFilter (QObject *, QEvent *)

Event filter.

Protected Member Functions

- virtual void widgetMousePressEvent (QMouseEvent *)
- virtual void widgetMouseReleaseEvent (QMouseEvent *)
- virtual void widgetMouseMoveEvent (QMouseEvent *)
- virtual void widgetKeyPressEvent (QKeyEvent *)
- virtual void widgetKeyReleaseEvent (QKeyEvent *)
- virtual void paintEvent (QPaintEvent *)

Paint event

· virtual QBitmap contentsMask () const

Calculate a mask for the contents of the panned widget.

· virtual QPixmap grab () const

12.53.1 Detailed Description

QwtPanner provides panning of a widget.

QwtPanner grabs the contents of a widget, that can be dragged in all directions. The offset between the start and the end position is emitted by the panned signal.

QwtPanner grabs the content of the widget into a pixmap and moves the pixmap around, without initiating any repaint events for the widget. Areas, that are not part of content are not painted while panning. This makes panning fast enough for widgets, where repaints are too slow for mouse movements.

For widgets, where repaints are very fast it might be better to implement panning manually by mapping mouse events into paint events.

12.53.2 Constructor & Destructor Documentation

12.53.2.1 QwtPanner::QwtPanner (QWidget * parent)

Creates an panner that is enabled for the left mouse button.

Parameters

parent Parent widget to be panned

12.53.3 Member Function Documentation

12.53.3.1 QBitmap QwtPanner::contentsMask()const [protected], [virtual]

Calculate a mask for the contents of the panned widget.

Sometimes only parts of the contents of a widget should be panned. F.e. for a widget with a styled background with rounded borders only the area inside of the border should be panned.

Returns

An empty bitmap, indicating no mask

Reimplemented in QwtPlotPanner.

12.53.3.2 const QCursor QwtPanner::cursor () const

Returns

Cursor that is active while panning

```
See Also
```

setCursor()

12.53.3.3 bool QwtPanner::eventFilter (QObject * object, QEvent * event) [virtual]

Event filter.

When isEnabled() is true mouse events of the observed widget are filtered.

Parameters

| object | Object to be filtered |
|--------|-----------------------|
| event | Event |

Returns

Always false, beside for paint events for the parent widget.

See Also

widgetMousePressEvent(), widgetMouseReleaseEvent(), widgetMouseMoveEvent()

12.53.3.4 QPixmap QwtPanner::grab () const [protected], [virtual]

Grab the widget into a pixmap.

Returns

Grabbed pixmap

Reimplemented in QwtPlotPanner.

12.53.3.5 bool QwtPanner::isEnabled () const

Returns

true when enabled, false otherwise

See Also

setEnabled, eventFilter()

12.53.3.6 bool QwtPanner::isOrientationEnabled (Qt::Orientation o) const

Returns

True if an orientation is enabled

See Also

orientations(), setOrientations()

12.53.3.7 void QwtPanner::moved (int dx, int dy) [signal]

Signal emitted, while the widget moved, but panning is not finished.

Parameters

| dx | Offset in horizontal direction |
|----|--------------------------------|
| dy | Offset in vertical direction |

12.53.3.8 void QwtPanner::paintEvent (QPaintEvent * pe) [protected], [virtual]

Paint event.

Repaint the grabbed pixmap on its current position and fill the empty spaces by the background of the parent widget.

Parameters 2 4 1

| pe | Paint event |
|----|-------------|

12.53.3.9 void QwtPanner::panned (int dx, int dy) [signal]

Signal emitted, when panning is done

Parameters

| dx | Offset in horizontal direction |
|----|--------------------------------|
| dy | Offset in vertical direction |

12.53.3.10 void QwtPanner::setAbortKey (int key, Qt::KeyboardModifiers modifiers = Qt::NoModifier)

Change the abort key The defaults are Qt::Key_Escape and Qt::NoModifiers

Parameters

| key | Key (See Qt::Keycode) |
|-----------|-------------------------|
| modifiers | Keyboard modifiers |

12.53.3.11 void QwtPanner::setCursor (const QCursor & cursor)

Change the cursor, that is active while panning The default is the cursor of the parent widget.

Parameters

| cursor | New cursor |
|--------|------------|

See Also

setCursor()

12.53.3.12 void QwtPanner::setEnabled (bool on)

En/disable the panner.

When enabled is true an event filter is installed for the observed widget, otherwise the event filter is removed.

Parameters

| on | true or false |
|----|---------------|

See Also

isEnabled(), eventFilter()

12.53.3.13 void QwtPanner::setMouseButton (Qt::MouseButton button, Qt::KeyboardModifiers modifiers = Qt::NoModifier)

Change the mouse button and modifiers used for panning The defaults are Qt::LeftButton and Qt::NoModifier

12.53.3.14 void QwtPanner::setOrientations (Qt::Orientations o)

Set the orientations, where panning is enabled The default value is in both directions: Qt::Horizontal | Qt::Vertical /param o Orientation

12.53.3.15 void QwtPanner::widgetKeyPressEvent (QKeyEvent * keyEvent) [protected], [virtual]

Handle a key press event for the observed widget.

Parameters

keyEvent Key event

See Also

eventFilter(), widgetKeyReleaseEvent()

12.53.3.16 void QwtPanner::widgetKeyReleaseEvent (QKeyEvent * keyEvent) [protected], [virtual]

Handle a key release event for the observed widget.

Parameters

keyEvent Key event

See Also

eventFilter(), widgetKeyReleaseEvent()

12.53.3.17 void QwtPanner::widgetMouseMoveEvent (QMouseEvent * mouseEvent) [protected], [virtual]

Handle a mouse move event for the observed widget.

Parameters

mouseEvent Mouse event

See Also

eventFilter(), widgetMousePressEvent(), widgetMouseReleaseEvent()

12.53.3.18 void QwtPanner::widgetMousePressEvent (QMouseEvent * mouseEvent) [protected], [virtual]

Handle a mouse press event for the observed widget.

Parameters

mouseEvent Mouse event

See Also

eventFilter(), widgetMouseReleaseEvent(), widgetMouseMoveEvent(),

12.53.3.19 void QwtPanner::widgetMouseReleaseEvent (QMouseEvent * mouseEvent) [protected], [virtual]

Handle a mouse release event for the observed widget.

Parameters

| mouseEvent | Mouse event |
|------------|-------------|
|------------|-------------|

See Also

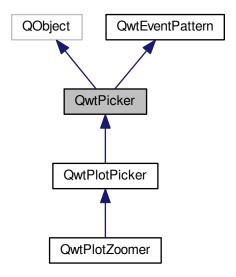
eventFilter(), widgetMousePressEvent(), widgetMouseMoveEvent(),

12.54 QwtPicker Class Reference

QwtPicker provides selections on a widget.

```
#include <qwt_picker.h>
```

Inheritance diagram for QwtPicker:



Public Types

- enum RubberBand {
 NoRubberBand = 0, HLineRubberBand, VLineRubberBand, CrossRubberBand,
 RectRubberBand, EllipseRubberBand, PolygonRubberBand, UserRubberBand = 100 }
- enum DisplayMode { AlwaysOff, AlwaysOn, ActiveOnly }
 Display mode.
- enum ResizeMode { Stretch, KeepSize }

Public Slots

• void setEnabled (bool)

En/disable the picker.

Signals

- void activated (bool on)
- void selected (const QPolygon &polygon)
- void appended (const QPoint &pos)
- void moved (const QPoint &pos)
- · void removed (const QPoint &pos)
- · void changed (const QPolygon &selection)

Public Member Functions

- QwtPicker (QWidget *parent)
- QwtPicker (RubberBand rubberBand, DisplayMode trackerMode, QWidget *)
- virtual ~QwtPicker ()

Destructor.

- void setStateMachine (QwtPickerMachine *)
- const QwtPickerMachine * stateMachine () const
- QwtPickerMachine * stateMachine ()
- void setRubberBand (RubberBand)
- · RubberBand rubberBand () const
- void setTrackerMode (DisplayMode)

Set the display mode of the tracker.

- · DisplayMode trackerMode () const
- void setResizeMode (ResizeMode)

Set the resize mode.

- ResizeMode resizeMode () const
- void setRubberBandPen (const QPen &)
- QPen rubberBandPen () const
- · void setTrackerPen (const QPen &)
- QPen trackerPen () const
- void setTrackerFont (const QFont &)
- · QFont trackerFont () const
- bool isEnabled () const
- bool isActive () const
- virtual bool eventFilter (QObject *, QEvent *)

Event filter.

QWidget * parentWidget ()

Return the parent widget, where the selection happens.

const QWidget * parentWidget () const

Return the parent widget, where the selection happens.

- · virtual QPainterPath pickArea () const
- virtual void drawRubberBand (QPainter *) const
- virtual void drawTracker (QPainter *) const
- virtual QRegion rubberBandMask () const
- virtual QwtText trackerText (const QPoint &pos) const

Return the label for a position.

- · QPoint trackerPosition () const
- virtual QRect trackerRect (const QFont &) const
- QPolygon selection () const

Protected Member Functions

• virtual QPolygon adjustedPoints (const QPolygon &) const

Map the pickedPoints() into a selection()

- virtual void transition (const QEvent *)
- virtual void begin ()
- virtual void append (const QPoint &)
- virtual void move (const QPoint &)
- · virtual void remove ()
- virtual bool end (bool ok=true)

Close a selection setting the state to inactive.

· virtual bool accept (QPolygon &) const

Validate and fix up the selection.

- · virtual void reset ()
- virtual void widgetMousePressEvent (QMouseEvent *)
- virtual void widgetMouseReleaseEvent (QMouseEvent *)
- virtual void widgetMouseDoubleClickEvent (QMouseEvent *)
- virtual void widgetMouseMoveEvent (QMouseEvent *)
- virtual void widgetWheelEvent (QWheelEvent *)
- virtual void widgetKeyPressEvent (QKeyEvent *)
- virtual void widgetKeyReleaseEvent (QKeyEvent *)
- virtual void widgetEnterEvent (QEvent *)
- virtual void widgetLeaveEvent (QEvent *)
- virtual void stretchSelection (const QSize &oldSize, const QSize &newSize)
- virtual void updateDisplay ()

Update the state of rubber band and tracker label.

- const QwtWidgetOverlay * rubberBandOverlay () const
- const QwtWidgetOverlay * trackerOverlay () const
- const QPolygon & pickedPoints () const

12.54.1 Detailed Description

QwtPicker provides selections on a widget.

QwtPicker filters all enter, leave, mouse and keyboard events of a widget and translates them into an array of selected points.

The way how the points are collected depends on type of state machine that is connected to the picker. Qwt offers a couple of predefined state machines for selecting:

· Nothing

QwtPickerTrackerMachine

· Single points

QwtPickerClickPointMachine, QwtPickerDragPointMachine

· Rectangles

QwtPickerClickRectMachine, QwtPickerDragRectMachine

· Polygons

QwtPickerPolygonMachine

While these state machines cover the most common ways to collect points it is also possible to implement individual machines as well.

QwtPicker translates the picked points into a selection using the adjustedPoints() method. adjustedPoints() is intended to be reimplemented to fix up the selection according to application specific requirements. (F.e. when an application accepts rectangles of a fixed aspect ratio only.)

Optionally QwtPicker support the process of collecting points by a rubber band and tracker displaying a text for the current mouse position.

Example

```
#include <qwt_picker.h>
#include <qwt_picker_machine.h>

QwtPicker *picker = new QwtPicker(widget);
picker->setStateMachine(new QwtPickerDragRectMachine);
picker->setTrackerMode(QwtPicker::ActiveOnly);
picker->setRubberBand(QwtPicker::RectRubberBand);
```

The state machine triggers the following commands:

begin()

Activate/Initialize the selection.

append()

Add a new point

• move()

Change the position of the last point.

• remove()

Remove the last point.

• end()

Terminate the selection and call accept to validate the picked points.

The picker is active (isActive()), between begin() and end(). In active state the rubber band is displayed, and the tracker is visible in case of trackerMode is ActiveOnly or AlwaysOn.

The cursor can be moved using the arrow keys. All selections can be aborted using the abort key. (QwtEvent-Pattern::KeyPatternCode)

Warning

In case of QWidget::NoFocus the focus policy of the observed widget is set to QWidget::WheelFocus and mouse tracking will be manipulated while the picker is active, or if trackerMode() is AlwayOn.

```
12.54.2 Member Enumeration Documentation
```

12.54.2.1 enum QwtPicker::DisplayMode

Display mode.

See Also

setTrackerMode(), trackerMode(), isActive()

Enumerator

```
AlwaysOff Display never.AlwaysOn Display always.ActiveOnly Display only when the selection is active.
```

```
12.54.2.2 enum QwtPicker::ResizeMode
```

Controls what to do with the selected points of an active selection when the observed widget is resized.

The default value is QwtPicker::Stretch.

See Also

setResizeMode()

Enumerator

Stretch All points are scaled according to the new size,.

KeepSize All points remain unchanged.

12.54.2.3 enum QwtPicker::RubberBand

Rubber band style

The default value is QwtPicker::NoRubberBand.

See Also

setRubberBand(), rubberBand()

Enumerator

NoRubberBand No rubberband.

HLineRubberBand A horizontal line (only for QwtPickerMachine::PointSelection)

VLineRubberBand A vertical line (only for QwtPickerMachine::PointSelection)

CrossRubberBand A crosshair (only for QwtPickerMachine::PointSelection)

RectRubberBand A rectangle (only for QwtPickerMachine::RectSelection)

EllipseRubberBand An ellipse (only for QwtPickerMachine::RectSelection)

PolygonRubberBand A polygon (only for QwtPickerMachine::PolygonSelection)

UserRubberBand Values >= UserRubberBand can be used to define additional rubber bands.

12.54.3 Constructor & Destructor Documentation

```
12.54.3.1 QwtPicker::QwtPicker(QWidget* parent) [explicit]
```

Constructor

Creates an picker that is enabled, but without a state machine. rubber band and tracker are disabled.

Parameters

parent Parent widget, that will be observed

12.54.3.2 QwtPicker::QwtPicker (RubberBand rubberBand, DisplayMode trackerMode, QWidget * parent) [explicit]

Constructor

Parameters

Generated on Thu Sep 18 2014 16:59:51 for Qwt User's Guide by Doxygen

| rubberBand | Rubber band style |
|-------------|--------------------------------------|
| trackerMode | Tracker mode |
| parent | Parent widget, that will be observed |

12.54.4 Member Function Documentation

```
12.54.4.1 bool QwtPicker::accept ( QPolygon & selection ) const [protected], [virtual]
```

Validate and fix up the selection.

Accepts all selections unmodified

Parameters

| selection | Selection to validate and fix up |
|-----------|----------------------------------|
|-----------|----------------------------------|

Returns

true, when accepted, false otherwise

Reimplemented in QwtPlotZoomer.

```
12.54.4.2 void QwtPicker::activated (bool on ) [signal]
```

A signal indicating, when the picker has been activated. Together with setEnabled() it can be used to implement selections with more than one picker.

Parameters

```
on True, when the picker has been activated
```

12.54.4.3 QPolygon QwtPicker::adjustedPoints (const QPolygon & points) const [protected], [virtual]

Map the pickedPoints() into a selection()

adjustedPoints() maps the points, that have been collected on the parentWidget() into a selection(). The default implementation simply returns the points unmodified.

The reason, why a selection() differs from the picked points depends on the application requirements. F.e.:

- A rectangular selection might need to have a specific aspect ratio only.
- · A selection could accept non intersecting polygons only.
- ...

The example below is for a rectangular selection, where the first point is the center of the selected rectangle.

Example

```
QPolygon MyPicker::adjustedPoints(const QPolygon &points) const
{
    QPolygon adjusted;
    if ( points.size() == 2 )
    {
        const int width = qAbs(points[1].x() - points[0].x());
        const int height = qAbs(points[1].y() - points[0].y());

        QRect rect(0, 0, 2 * width, 2 * height);
        rect.moveCenter(points[0]);

        adjusted += rect.topLeft();
        adjusted += rect.bottomRight();
    }
    return adjusted;
}
```

Parameters

points | Selected points

Returns

Selected points unmodified

12.54.4.4 void QwtPicker::append (const QPoint & pos) [protected], [virtual]

Append a point to the selection and update rubber band and tracker. The appended() signal is emitted.

Parameters

pos Additional point

See Also

isActive(), begin(), end(), move(), appended()

Reimplemented in QwtPlotPicker.

12.54.4.5 void QwtPicker::appended (const QPoint & pos) [signal]

A signal emitted when a point has been appended to the selection

Parameters

pos Position of the appended point.

See Also

append(). moved()

12.54.4.6 void QwtPicker::begin () [protected], [virtual]

Open a selection setting the state to active

See Also

isActive(), end(), append(), move()

Reimplemented in QwtPlotZoomer.

12.54.4.7 void QwtPicker::changed (const QPolygon & selection) [signal]

A signal emitted when the active selection has been changed. This might happen when the observed widget is resized.

Parameters

selection Changed selection

See Also

stretchSelection()

12.54.4.8 void QwtPicker::drawRubberBand (QPainter * painter) const [virtual]

Draw a rubber band, depending on rubberBand()

| painter | Painter, initialized with a clip region |
|---------|---|
|---------|---|

See Also

rubberBand(), RubberBand

12.54.4.9 void QwtPicker::drawTracker (QPainter * painter) const [virtual]

Draw the tracker

Parameters

| painter | Painter |
|---------|---------|

See Also

trackerRect(), trackerText()

12.54.4.10 bool QwtPicker::end (bool ok = true) [protected], [virtual]

Close a selection setting the state to inactive.

The selection is validated and maybe fixed by accept().

Parameters

ok If true, complete the selection and emit a selected signal otherwise discard the selection.

Returns

true if the selection is accepted, false otherwise

See Also

isActive(), begin(), append(), move(), selected(), accept()

Reimplemented in QwtPlotZoomer, and QwtPlotPicker.

12.54.4.11 bool QwtPicker::eventFilter (QObject * object, QEvent * event) [virtual]

Event filter.

When isEnabled() is true all events of the observed widget are filtered. Mouse and keyboard events are translated into widgetMouse- and widgetKey- and widgetWheel-events. Paint and Resize events are handled to keep rubber band and tracker up to date.

Parameters

| obje | ct Object to be filtered |
|------|--------------------------|
| eve | |

Returns

Always false.

See Also

widgetEnterEvent(), widgetLeaveEvent(), widgetMousePressEvent(), widgetMouseReleaseEvent(), widgetMouseDoubleClickEvent(), widgetMouseMoveEvent(), widgetWheelEvent(), widgetKeyPressEvent(), widgetKeyReleaseEvent(), QObject::installEventFilter(), QObject::event()

```
12.54.4.12 bool QwtPicker::isActive ( ) const
A picker is active between begin() and end().
Returns
     true if the selection is active.
12.54.4.13 bool QwtPicker::isEnabled ( ) const
Returns
      true when enabled, false otherwise
See Also
      setEnabled(), eventFilter()
12.54.4.14 void QwtPicker::move (const QPoint & pos) [protected], [virtual]
Move the last point of the selection The moved() signal is emitted.
Parameters
               pos
                     New position
See Also
      isActive(), begin(), end(), append()
Reimplemented in QwtPlotPicker.
12.54.4.15 void QwtPicker::moved (const QPoint & pos) [signal]
A signal emitted whenever the last appended point of the selection has been moved.
Parameters
                     Position of the moved last point of the selection.
               pos
See Also
      move(), appended()
12.54.4.16 QPainterPath QwtPicker::pickArea ( ) const [virtual]
Find the area of the observed widget, where selection might happen.
Returns
      parentWidget()->contentsRect()
12.54.4.17 const QPolygon & QwtPicker::pickedPoints ( ) const [protected]
Return the points, that have been collected so far. The selection() is calculated from the pickedPoints() in adjusted-
Points().
Returns
```

Picked points

```
12.54.4.18 void QwtPicker::remove() [protected], [virtual]
Remove the last point of the selection The removed() signal is emitted.
See Also
     isActive(), begin(), end(), append(), move()
12.54.4.19 void QwtPicker::removed (const QPoint & pos) [signal]
A signal emitted whenever the last appended point of the selection has been removed.
Parameters
              pos | Position of the point, that has been removed
See Also
     remove(), appended()
12.54.4.20 void QwtPicker::reset() [protected], [virtual]
Reset the state machine and terminate (end(false)) the selection
12.54.4.21 QwtPicker::ResizeMode QwtPicker::resizeMode ( ) const
Returns
     Resize mode
See Also
     setResizeMode(), ResizeMode
12.54.4.22 QwtPicker::RubberBand QwtPicker::rubberBand ( ) const
Returns
     Rubber band style
See Also
     setRubberBand(), RubberBand, rubberBandPen()
12.54.4.23 QRegion QwtPicker::rubberBandMask( )const [virtual]
Calculate the mask for the rubber band overlay
Returns
     Region for the mask
See Also
     QWidget::setMask()
12.54.4.24 const QwtWidgetOverlay * QwtPicker::rubberBandOverlay ( ) const [protected]
Returns
     Overlay displaying the rubber band
```

12.54.4.25 QPen QwtPicker::rubberBandPen () const

Returns

Rubber band pen

See Also

setRubberBandPen(), rubberBand()

12.54.4.26 void QwtPicker::selected (const QPolygon & polygon) [signal]

A signal emitting the selected points, at the end of a selection.

Parameters

polygon | Selected points

12.54.4.27 QPolygon QwtPicker::selection () const

Returns

Selected points

See Also

pickedPoints(), adjustedPoints()

12.54.4.28 void QwtPicker::setEnabled (bool enabled) [slot]

En/disable the picker.

When enabled is true an event filter is installed for the observed widget, otherwise the event filter is removed.

Parameters

enabled true or false

See Also

isEnabled(), eventFilter()

12.54.4.29 void QwtPicker::setResizeMode (ResizeMode mode)

Set the resize mode.

The resize mode controls what to do with the selected points of an active selection when the observed widget is resized.

Stretch means the points are scaled according to the new size, KeepSize means the points remain unchanged.

The default mode is Stretch.

Parameters

mode Resize mode

See Also

resizeMode(), ResizeMode

12.54.4.30 void QwtPicker::setRubberBand (RubberBand rubberBand)

Set the rubber band style

rubberBand Rubber band style The default value is NoRubberBand.

See Also

rubberBand(), RubberBand, setRubberBandPen()

12.54.4.31 void QwtPicker::setRubberBandPen (const QPen & pen)

Set the pen for the rubberband

Parameters

pen Rubber band pen

See Also

rubberBandPen(), setRubberBand()

12.54.4.32 void QwtPicker::setStateMachine (QwtPickerMachine * stateMachine)

Set a state machine and delete the previous one

Parameters

stateMachine State machine

See Also

stateMachine()

12.54.4.33 void QwtPicker::setTrackerFont (const QFont & font)

Set the font for the tracker

Parameters

font Tracker font

See Also

trackerFont(), setTrackerMode(), setTrackerPen()

12.54.4.34 void QwtPicker::setTrackerMode (DisplayMode mode)

Set the display mode of the tracker.

A tracker displays information about current position of the cursor as a string. The display mode controls if the tracker has to be displayed whenever the observed widget has focus and cursor (AlwaysOn), never (AlwaysOff), or only when the selection is active (ActiveOnly).

Parameters

mode Tracker display mode

Warning

In case of AlwaysOn, mouseTracking will be enabled for the observed widget.

See Also

trackerMode(), DisplayMode

12.54.4.35 void QwtPicker::setTrackerPen (const QPen & pen)

Set the pen for the tracker

| pen | Tracker pen |
|-----|-------------|
|-----|-------------|

See Also

trackerPen(), setTrackerMode(), setTrackerFont()

12.54.4.36 const QwtPickerMachine * QwtPicker::stateMachine () const

Returns

Assigned state machine

See Also

setStateMachine()

12.54.4.37 QwtPickerMachine * QwtPicker::stateMachine ()

Returns

Assigned state machine

See Also

setStateMachine()

12.54.4.38 void QwtPicker::stretchSelection (const QSize & oldSize, const QSize & newSize) [protected], [virtual]

Scale the selection by the ratios of oldSize and newSize The changed() signal is emitted.

Parameters

| oldSize | Previous size |
|---------|---------------|
| newSize | Current size |

See Also

ResizeMode, setResizeMode(), resizeMode()

12.54.4.39 QFont QwtPicker::trackerFont () const

Returns

Tracker font

See Also

setTrackerFont(), trackerMode(), trackerPen()

12.54.4.40 QwtPicker::DisplayMode QwtPicker::trackerMode () const

Returns

Tracker display mode

See Also

setTrackerMode(), DisplayMode

12.54.4.41 const QwtWidgetOverlay * QwtPicker::trackerOverlay () const [protected]

Returns

Overlay displaying the tracker text

12.54.4.42 QPen QwtPicker::trackerPen () const

Returns

Tracker pen

See Also

setTrackerPen(), trackerMode(), trackerFont()

12.54.4.43 QPoint QwtPicker::trackerPosition () const

Returns

Current position of the tracker

12.54.4.44 QRect QwtPicker::trackerRect (const QFont & font) const [virtual]

Calculate the bounding rectangle for the tracker text from the current position of the tracker

Parameters

| f | Fort of the dispersion to de |
|------|------------------------------|
| font | Font of the tracker text |

Returns

Bounding rectangle of the tracker text

See Also

trackerPosition()

12.54.4.45 QwtText QwtPicker::trackerText (const QPoint & pos) const [virtual]

Return the label for a position.

In case of HLineRubberBand the label is the value of the y position, in case of VLineRubberBand the value of the x position. Otherwise the label contains x and y position separated by a \ddot{y} .

The format for the string conversion is "%d".

Parameters

| pos | Position |
|-----|----------|

Returns

Converted position as string

Reimplemented in QwtPlotPicker.

12.54.4.46 void QwtPicker::transition (const QEvent * event) [protected], [virtual]

Passes an event to the state machine and executes the resulting commands. Append and Move commands use the current position of the cursor (QCursor::pos()).

| event | Event |
|-------|-------|
|-------|-------|

12.54.4.47 void QwtPicker::widgetEnterEvent (QEvent * **event**) [protected], [virtual]

Handle a enter event for the observed widget.

Parameters

| event | Qt event |
|-------|----------|
|-------|----------|

See Also

eventFilter(), widgetMousePressEvent(), widgetMouseReleaseEvent(), widgetMouseDoubleClickEvent(), widgetWheelEvent(), widgetKeyPressEvent(), widgetKeyPressEvent()

12.54.4.48 void QwtPicker::widgetKeyPressEvent (QKeyEvent * keyEvent) [protected], [virtual]

Handle a key press event for the observed widget.

Selections can be completely done by the keyboard. The arrow keys move the cursor, the abort key aborts a selection. All other keys are handled by the current state machine.

Parameters

| kev | Event | Key event |
|------|--------|-----------|
| ney. | LVCIII | ney event |

See Also

 $eventFilter(), \quad widgetMousePressEvent(), \quad widgetMouseReleaseEvent(), \quad widgetMouseDoubleClickEvent(), \\ widgetMouseMoveEvent(), \quad widgetWheelEvent(), \quad widgetKeyReleaseEvent(), \quad stateMachine(), \quad QwtEvent-Pattern::KeyPatternCode$

Reimplemented in QwtPlotZoomer.

12.54.4.49 void QwtPicker::widgetKeyReleaseEvent (QKeyEvent * keyEvent) [protected], [virtual]

Handle a key release event for the observed widget.

Passes the event to the state machine.

Parameters

| keyEvent | Key event |
|----------|-----------|
|----------|-----------|

See Also

eventFilter(), widgetMousePressEvent(), widgetMouseReleaseEvent(), widgetMouseDoubleClickEvent(), widgetMouseMoveEvent(), widgetWheelEvent(), widgetKeyPressEvent(), stateMachine()

12.54.4.50 void QwtPicker::widgetLeaveEvent(QEvent * event) [protected], [virtual]

Handle a leave event for the observed widget.

Parameters

| event | Qt event |
|-------|----------|

See Also

 $eventFilter(), \quad widgetMousePressEvent(), \quad widgetMouseReleaseEvent(), \quad widgetMouseDoubleClickEvent(), \\ widgetWheelEvent(), \quad widgetKeyPressEvent(), \quad widgetKeyPressEvent(), \\ widgetMousePressEvent(), \quad widgetMousePressEvent(), \\ widgetMousePressEvent(), \quad widgetMousePressEvent(), \\ widgetMousePressEvent(), \quad widgetMousePressEvent(), \\ wid$

```
12.54.4.51 void QwtPicker::widgetMouseDoubleClickEvent ( QMouseEvent * mouseEvent ) [protected], [virtual]
```

Handle mouse double click event for the observed widget.

| mouseEvent | Mouse event |
|------------|-------------|
|------------|-------------|

See Also

eventFilter(), widgetMousePressEvent(), widgetMouseReleaseEvent(), widgetMouseMoveEvent(), widgetWheelEvent(), widgetKeyPressEvent(), widgetKeyPressEvent()

12.54.4.52 void QwtPicker::widgetMouseMoveEvent (QMouseEvent * mouseEvent) [protected], [virtual]

Handle a mouse move event for the observed widget.

Parameters

| mouseEvent | Mouse event |
|------------|-------------|
| | |

See Also

eventFilter(), widgetMousePressEvent(), widgetMouseReleaseEvent(), widgetMouseDoubleClickEvent(), widgetWheelEvent(), widgetKeyPressEvent(), widgetKeyPressEvent()

12.54.4.53 void QwtPicker::widgetMousePressEvent (QMouseEvent * mouseEvent) [protected], [virtual]

Handle a mouse press event for the observed widget.

Parameters

| mouseEvent | Mouse event |
|------------|-------------|
|------------|-------------|

See Also

 $eventFilter(), \quad widgetMouseReleaseEvent(), \quad widgetMouseDoubleClickEvent(), \quad widgetMouseMoveEvent(), \\ widgetWheelEvent(), \quad widgetKeyPressEvent(), \quad widgetKeyReleaseEvent(), \\ widgetMouseMoveEvent(), \quad widgetMouseMoveEvent(), \\ widgetMoveEvent(), \\ widgetMouseMoveEvent(), \\ widgetMouseMoveEvent(),$

12.54.4.54 void QwtPicker::widgetMouseReleaseEvent(QMouseEvent * mouseEvent) [protected], [virtual]

Handle a mouse release event for the observed widget.

Parameters

| mouseEvent | Mouse event |
|------------|-------------|

See Also

eventFilter(), widgetMousePressEvent(), widgetMouseDoubleClickEvent(), widgetMouseMoveEvent(), widgetWheelEvent(), widgetKeyPressEvent(), widgetKeyPressEvent(

Reimplemented in QwtPlotZoomer.

12.54.4.55 void QwtPicker::widgetWheelEvent (QWheelEvent * wheelEvent) [protected], [virtual]

Handle a wheel event for the observed widget.

Move the last point of the selection in case of isActive() == true

Parameters

| wheelEvent | Wheel event |
|------------|-------------|
|------------|-------------|

See Also

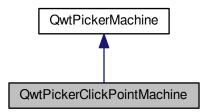
 $eventFilter(), \quad widgetMousePressEvent(), \quad widgetMouseReleaseEvent(), \quad widgetMouseDoubleClickEvent(), \\ widgetMouseMoveEvent(), \quad widgetKeyPressEvent(), \quad widgetKeyPressEvent(), \\ widgetMouseMoveEvent(), \quad widgetKeyPressEvent(), \\ widgetMouseMoveEvent(), \quad widgetMousePressEvent(), \\ widgetMouseMoveEvent(), \quad widgetMousePressEvent(), \\ widgetMouseMoveEvent(), \\ widgetMoveEvent(), \\ widgetMoveEvent()$

12.55 QwtPickerClickPointMachine Class Reference

A state machine for point selections.

```
#include <qwt_picker_machine.h>
```

Inheritance diagram for QwtPickerClickPointMachine:



Public Member Functions

• QwtPickerClickPointMachine ()

Constructor.

virtual QList < Command > transition (const QwtEventPattern &, const QEvent *)
 Transition.

Additional Inherited Members

12.55.1 Detailed Description

A state machine for point selections.

Pressing QwtEventPattern::MouseSelect1 or QwtEventPattern::KeySelect1 selects a point.

See Also

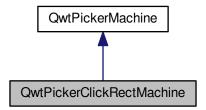
QwtEventPattern::MousePatternCode, QwtEventPattern::KeyPatternCode

12.56 QwtPickerClickRectMachine Class Reference

A state machine for rectangle selections.

#include <qwt_picker_machine.h>

Inheritance diagram for QwtPickerClickRectMachine:



Public Member Functions

• QwtPickerClickRectMachine ()

Constructor.

virtual QList< Command > transition (const QwtEventPattern &, const QEvent *)

Transition.

Additional Inherited Members

12.56.1 Detailed Description

A state machine for rectangle selections.

Pressing QwtEventPattern::MouseSelect1 starts the selection, releasing it selects the first point. Pressing it again selects the second point and terminates the selection. Pressing QwtEventPattern::KeySelect1 also starts the selection, a second press selects the first point. A third one selects the second point and terminates the selection.

See Also

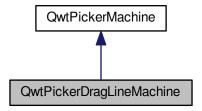
QwtEventPattern::MousePatternCode, QwtEventPattern::KeyPatternCode

12.57 QwtPickerDragLineMachine Class Reference

A state machine for line selections.

#include <qwt_picker_machine.h>

Inheritance diagram for QwtPickerDragLineMachine:



Public Member Functions

QwtPickerDragLineMachine ()

Constructor.

virtual QList< Command > transition (const QwtEventPattern &, const QEvent *)

Transition.

Additional Inherited Members

12.57.1 Detailed Description

A state machine for line selections.

Pressing QwtEventPattern::MouseSelect1 selects the first point, releasing it the second point. Pressing QwtEvent-Pattern::KeySelect1 also selects the first point, a second press selects the second point and terminates the selection.

A common use case of QwtPickerDragLineMachine are pickers for distance measurements.

See Also

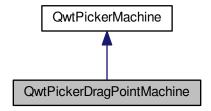
QwtEventPattern::MousePatternCode, QwtEventPattern::KeyPatternCode

12.58 QwtPickerDragPointMachine Class Reference

A state machine for point selections.

#include <qwt_picker_machine.h>

Inheritance diagram for QwtPickerDragPointMachine:



Public Member Functions

· QwtPickerDragPointMachine ()

Constructor.

virtual QList< Command > transition (const QwtEventPattern &, const QEvent *)
 Transition.

Additional Inherited Members

12.58.1 Detailed Description

A state machine for point selections.

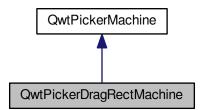
Pressing QwtEventPattern::MouseSelect1 or QwtEventPattern::KeySelect1 starts the selection, releasing QwtEventPattern::MouseSelect1 or a second press of QwtEventPattern::KeySelect1 terminates it.

12.59 QwtPickerDragRectMachine Class Reference

A state machine for rectangle selections.

```
#include <qwt_picker_machine.h>
```

Inheritance diagram for QwtPickerDragRectMachine:



Public Member Functions

• QwtPickerDragRectMachine ()

Constructor.

virtual QList < Command > transition (const QwtEventPattern &, const QEvent *)
 Transition.

Additional Inherited Members

12.59.1 Detailed Description

A state machine for rectangle selections.

Pressing QwtEventPattern::MouseSelect1 selects the first point, releasing it the second point. Pressing QwtEvent-Pattern::KeySelect1 also selects the first point, a second press selects the second point and terminates the selection.

See Also

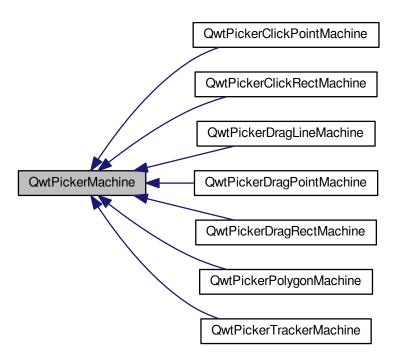
QwtEventPattern::MousePatternCode, QwtEventPattern::KeyPatternCode

12.60 QwtPickerMachine Class Reference

A state machine for **QwtPicker** selections.

#include <qwt_picker_machine.h>

Inheritance diagram for QwtPickerMachine:



Public Types

- enum SelectionType { NoSelection = -1, PointSelection, RectSelection, PolygonSelection }
- enum Command {

Begin, Append, Move, Remove, End }

Commands - the output of a state machine.

Public Member Functions

QwtPickerMachine (SelectionType)

Constructor.

virtual ~QwtPickerMachine ()

Destructor.

• virtual QList< Command > transition (const QwtEventPattern &, const QEvent *)=0

Transition.

· void reset ()

Set the current state to 0.

• int state () const

Return the current state.

void setState (int)

Change the current state.

SelectionType selectionType () const

Return the selection type.

12.60.1 Detailed Description

A state machine for **QwtPicker** selections.

QwtPickerMachine accepts key and mouse events and translates them into selection commands.

See Also

 $QwtEventPattern:: Mouse Pattern Code, \ QwtEventPattern:: Key Pattern Code \\$

12.60.2 Member Enumeration Documentation

12.60.2.1 enum QwtPickerMachine::SelectionType

Type of a selection.

See Also

selectionType()

Enumerator

NoSelection The state machine not usable for any type of selection.

PointSelection The state machine is for selecting a single point.

RectSelection The state machine is for selecting a rectangle (2 points).

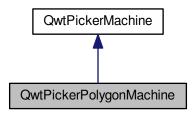
PolygonSelection The state machine is for selecting a polygon (many points).

12.61 QwtPickerPolygonMachine Class Reference

A state machine for polygon selections.

#include <qwt_picker_machine.h>

Inheritance diagram for QwtPickerPolygonMachine:



Public Member Functions

• QwtPickerPolygonMachine ()

Constructor.

virtual QList< Command > transition (const QwtEventPattern &, const QEvent *)

Transition.

Additional Inherited Members

12.61.1 Detailed Description

A state machine for polygon selections.

Pressing QwtEventPattern::MouseSelect1 or QwtEventPattern::KeySelect1 starts the selection and selects the first point, or appends a point. Pressing QwtEventPattern::MouseSelect2 or QwtEventPattern::KeySelect2 appends the last point and terminates the selection.

See Also

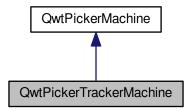
QwtEventPattern::MousePatternCode, QwtEventPattern::KeyPatternCode

12.62 QwtPickerTrackerMachine Class Reference

A state machine for indicating mouse movements.

#include <qwt_picker_machine.h>

Inheritance diagram for QwtPickerTrackerMachine:



Public Member Functions

• QwtPickerTrackerMachine ()

Constructor.

virtual QList< Command > transition (const QwtEventPattern &, const QEvent *)

Transition.

Additional Inherited Members

12.62.1 Detailed Description

A state machine for indicating mouse movements.

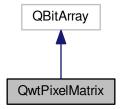
QwtPickerTrackerMachine supports displaying information corresponding to mouse movements, but is not intended for selecting anything. Begin/End are related to Enter/Leave events.

12.63 QwtPixelMatrix Class Reference

A bit field corresponding to the pixels of a rectangle.

#include <qwt_pixel_matrix.h>

Inheritance diagram for QwtPixelMatrix:



Public Member Functions

QwtPixelMatrix (const QRect &rect)

Constructor.

∼QwtPixelMatrix ()

Destructor.

- void setRect (const QRect &rect)
- QRect rect () const
- bool testPixel (int x, int y) const

Test if a pixel has been set.

• bool testAndSetPixel (int x, int y, bool on)

Set a pixel and test if a pixel has been set before.

• int index (int x, int y) const

Calculate the index in the bit field corresponding to a position.

12.63.1 Detailed Description

A bit field corresponding to the pixels of a rectangle.

QwtPixelMatrix is intended to filter out duplicates in an unsorted array of points.

12.63.2 Constructor & Destructor Documentation

12.63.2.1 QwtPixelMatrix::QwtPixelMatrix (const QRect & rect)

Constructor.

Parameters

| rect | Bounding rectangle for the matrix |
|------|-----------------------------------|

12.63.3 Member Function Documentation

12.63.3.1 int QwtPixelMatrix::index (int x, int y) const [inline]

Calculate the index in the bit field corresponding to a position.

| X | X-coordinate |
|---|--------------|
| У | Y-coordinate |

Returns

Index, when rect() contains pos - otherwise -1.

12.63.3.2 QRect QwtPixelMatrix::rect () const

Returns

Bounding rectangle

12.63.3.3 void QwtPixelMatrix::setRect (const QRect & rect)

Set the bounding rectangle of the matrix

Parameters

| rect | Bounding rectangle |
|------|--------------------|

Note

All bits are cleared

12.63.3.4 bool QwtPixelMatrix::testAndSetPixel(int x, int y, bool on) [inline]

Set a pixel and test if a pixel has been set before.

Parameters

| X | X-coordinate |
|----|---------------------|
| У | Y-coordinate |
| on | Set/Clear the pixel |

Returns

true, when pos is outside of rect(), or when the pixel was set before.

12.63.3.5 bool QwtPixelMatrix::testPixel (int x, int y) const [inline]

Test if a pixel has been set.

Parameters

| X | X-coordinate X-coordinate |
|---|---------------------------|
| у | Y-coordinate Y-coordinate |

Returns

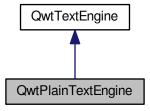
true, when pos is outside of rect(), or when the pixel has already been set.

12.64 QwtPlainTextEngine Class Reference

A text engine for plain texts.

#include <qwt_text_engine.h>

Inheritance diagram for QwtPlainTextEngine:



Public Member Functions

• QwtPlainTextEngine ()

Constructor.

virtual ~QwtPlainTextEngine ()

Destructor.

- · virtual double heightForWidth (const QFont &font, int flags, const QString &text, double width) const
- virtual QSizeF textSize (const QFont &font, int flags, const QString &text) const
- virtual void draw (QPainter *painter, const QRectF &rect, int flags, const QString &text) const Draw the text in a clipping rectangle.
- virtual bool mightRender (const QString &) const
- virtual void textMargins (const QFont &, const QString &, double &left, double &right, double &top, double &bottom) const

Additional Inherited Members

12.64.1 Detailed Description

A text engine for plain texts.

QwtPlainTextEngine renders texts using the basic Qt classes QPainter and QFontMetrics.

12.64.2 Member Function Documentation

12.64.2.1 void QwtPlainTextEngine::draw (QPainter * painter, const QRectF & rect, int flags, const QString & text) const [virtual]

Draw the text in a clipping rectangle.

A wrapper for QPainter::drawText.

Parameters

| painter | Painter |
|---------|--------------------|
| rect | Clipping rectangle |

| flags | Bitwise OR of the flags used like in QPainter::drawText |
|-------|---|
| text | Text to be rendered |

Implements QwtTextEngine.

12.64.2.2 double QwtPlainTextEngine::heightForWidth (const QFont & font, int flags, const QString & text, double width) const [virtual]

Find the height for a given width

Parameters

| font | Font of the text |
|-------|---|
| flags | Bitwise OR of the flags used like in QPainter::drawText |
| text | Text to be rendered |
| width | Width |

Returns

Calculated height

Implements QwtTextEngine.

12.64.2.3 bool QwtPlainTextEngine::mightRender (const QString &) const [virtual]

Test if a string can be rendered by this text engine.

Returns

Always true. All texts can be rendered by QwtPlainTextEngine

Implements QwtTextEngine.

12.64.2.4 void QwtPlainTextEngine::textMargins (const QFont & font, const QString & , double & left, double & right, double & top, double & bottom) const [virtual]

Return margins around the texts

Parameters

| font | Font of the text |
|--------|------------------------------------|
| left | Return 0 |
| right | Return 0 |
| top | Return value for the top margin |
| bottom | Return value for the bottom margin |

Implements QwtTextEngine.

12.64.2.5 QSizeF QwtPlainTextEngine::textSize(const QFont & font, int flags, const QString & text) const [virtual]

Returns the size, that is needed to render text

Parameters

| font | Font of the text |
|-------|---|
| flags | Bitwise OR of the flags used like in QPainter::drawText |
| text | Text to be rendered |

Returns

Caluclated size

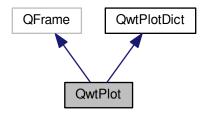
Implements QwtTextEngine.

12.65 QwtPlot Class Reference

A 2-D plotting widget.

```
#include <qwt_plot.h>
```

Inheritance diagram for QwtPlot:



Public Types

enum Axis { yLeft, yRight, xBottom, xTop, axisCnt }

Axis index.

• enum LegendPosition { LeftLegend, RightLegend, BottomLegend, TopLegend }

Public Slots

virtual void replot ()

Redraw the plot.

· void autoRefresh ()

Replots the plot if autoReplot() is true.

Signals

- void itemAttached (QwtPlotItem *plotItem, bool on)
- void legendDataChanged (const QVariant &itemInfo, const QList< QwtLegendData > &data)

Public Member Functions

QwtPlot (QWidget *=NULL)

Constructor.

QwtPlot (const QwtText &title, QWidget *=NULL)

Constructor.

virtual ~QwtPlot ()

Destructor.

- void applyProperties (const QString &)
- QString grabProperties () const
- void setAutoReplot (bool=true)

Set or reset the autoReplot option.

- bool autoReplot () const
- void setPlotLayout (QwtPlotLayout *)

Assign a new plot layout.

- QwtPlotLayout * plotLayout ()
- const QwtPlotLayout * plotLayout () const
- void setTitle (const QString &)
- void setTitle (const QwtText &t)
- QwtText title () const
- QwtTextLabel * titleLabel ()
- const QwtTextLabel * titleLabel () const
- void setFooter (const QString &)
- void setFooter (const QwtText &t)
- QwtText footer () const
- QwtTextLabel * footerLabel ()
- const QwtTextLabel * footerLabel () const
- void setCanvas (QWidget *)

Set the drawing canvas of the plot widget.

- QWidget * canvas ()
- const QWidget * canvas () const
- void setCanvasBackground (const QBrush &)

Change the background of the plotting area.

- QBrush canvasBackground () const
- virtual QwtScaleMap canvasMap (int axisId) const
- · double invTransform (int axisId, int pos) const
- · double transform (int axisId, double value) const

Transform a value into a coordinate in the plotting region.

- QwtScaleEngine * axisScaleEngine (int axisId)
- const QwtScaleEngine * axisScaleEngine (int axisId) const
- void setAxisScaleEngine (int axisId, QwtScaleEngine *)
- void setAxisAutoScale (int axisId, bool on=true)

Enable autoscaling for a specified axis.

- · bool axisAutoScale (int axisId) const
- void enableAxis (int axisId, bool tf=true)

Enable or disable a specified axis.

- · bool axisEnabled (int axisId) const
- void setAxisFont (int axisId, const QFont &f)

Change the font of an axis.

- · QFont axisFont (int axisId) const
- void setAxisScale (int axisId, double min, double max, double step=0)

Disable autoscaling and specify a fixed scale for a selected axis.

void setAxisScaleDiv (int axisId, const QwtScaleDiv &)

Disable autoscaling and specify a fixed scale for a selected axis.

void setAxisScaleDraw (int axisId, QwtScaleDraw *)

Set a scale draw.

double axisStepSize (int axisId) const

Return the step size parameter that has been set in setAxisScale.

· QwtInterval axisInterval (int axisId) const

Return the current interval of the specified axis.

const QwtScaleDiv & axisScaleDiv (int axisId) const

Return the scale division of a specified axis.

const QwtScaleDraw * axisScaleDraw (int axisId) const

Return the scale draw of a specified axis.

QwtScaleDraw * axisScaleDraw (int axisId)

Return the scale draw of a specified axis.

- const QwtScaleWidget * axisWidget (int axisId) const
- QwtScaleWidget * axisWidget (int axisId)
- void setAxisLabelAlignment (int axisId, Qt::Alignment)
- void setAxisLabelRotation (int axisId, double rotation)
- void setAxisTitle (int axisId, const QString &)

Change the title of a specified axis.

void setAxisTitle (int axisId, const QwtText &)

Change the title of a specified axis.

- QwtText axisTitle (int axisId) const
- void setAxisMaxMinor (int axisId, int maxMinor)
- · int axisMaxMinor (int axisId) const
- void setAxisMaxMajor (int axisId, int maxMajor)
- int axisMaxMajor (int axisId) const
- void insertLegend (QwtAbstractLegend *, LegendPosition=QwtPlot::RightLegend, double ratio=-1.0)

Insert a legend.

- QwtAbstractLegend * legend ()
- const QwtAbstractLegend * legend () const
- void updateLegend ()
- void updateLegend (const QwtPlotItem *)
- virtual QSize sizeHint () const
- virtual QSize minimumSizeHint () const

Return a minimum size hint.

virtual void updateLayout ()

Adjust plot content to its current size.

- virtual void drawCanvas (QPainter *)
- void updateAxes ()

Rebuild the axes scales.

· void updateCanvasMargins ()

Update the canvas margins.

 virtual void getCanvasMarginsHint (const QwtScaleMap maps[], const QRectF &canvasRect, double &left, double &top, double &right, double &bottom) const

Calculate the canvas margins.

virtual bool event (QEvent *)

Adds handling of layout requests.

virtual bool eventFilter (QObject *, QEvent *)

Event filter

- virtual void drawItems (QPainter *, const QRectF &, const QwtScaleMap maps[axisCnt]) const
- virtual QVariant itemToInfo (QwtPlotItem *) const

Build an information, that can be used to identify a plot item on the legend.

virtual QwtPlotItem * infoToItem (const QVariant &) const

Identify the plot item according to an item info object, that has bee generated from itemToInfo().

Protected Member Functions

virtual void resizeEvent (QResizeEvent *e)

Static Protected Member Functions

static bool axisValid (int axisId)

Friends

class QwtPlotItem

12.65.1 Detailed Description

A 2-D plotting widget.

QwtPlot is a widget for plotting two-dimensional graphs. An unlimited number of plot items can be displayed on its canvas. Plot items might be curves (QwtPlotCurve), markers (QwtPlotMarker), the grid (QwtPlotGrid), or anything else derived from QwtPlotItem. A plot can have up to four axes, with each plot item attached to an x- and a y axis. The scales at the axes can be explicitly set (QwtScaleDiv), or are calculated from the plot items, using algorithms (QwtScaleEngine) which can be configured separately for each axis.

The simpleplot example is a good starting point to see how to set up a plot widget.

Example

The following example shows (schematically) the most simple way to use QwtPlot. By default, only the left and bottom axes are visible and their scales are computed automatically.

```
#include <qwt_plot.h>
#include <qwt_plot_curve.h>

QwtPlot *myPlot = new QwtPlot("Two Curves", parent);

// add curves
QwtPlotCurve *curve1 = new QwtPlotCurve("Curve 1");
QwtPlotCurve *curve2 = new QwtPlotCurve("Curve 2");

// connect or copy the data to the curves
curve1->setData(...);
curve2->setData(...);

curve1->attach(myPlot);

curve2->attach(myPlot);

// finally, refresh the plot
myPlot->replot();
```

12.65.2 Member Enumeration Documentation

12.65.2.1 enum QwtPlot::Axis

Axis index.

Enumerator

```
yLeft Y axis left of the canvas.
yRight Y axis right of the canvas.
xBottom X axis below the canvas.
xTop X axis above the canvas.
axisCnt Number of axes.
```

12.65.2.2 enum QwtPlot::LegendPosition

Position of the legend, relative to the canvas.

See Also

insertLegend()

Enumerator

LeftLegend The legend will be left from the QwtPlot::yLeft axis.

RightLegend The legend will be right from the QwtPlot::yRight axis.

BottomLegend The legend will be below the footer.

TopLegend The legend will be above the title.

12.65.3 Constructor & Destructor Documentation

```
12.65.3.1 QwtPlot::QwtPlot(QWidget * parent = NULL) [explicit]
```

Constructor.

Parameters

| parent | Parent widget |
|--------|---------------|

12.65.3.2 QwtPlot::QwtPlot (const QwtText & title, QWidget * parent = NULL) [explicit]

Constructor.

Parameters

| title | Title text |
|--------|---------------|
| parent | Parent widget |

12.65.4 Member Function Documentation

12.65.4.1 void QwtPlot::applyProperties (const QString &)

This method is intended for manipulating the plot widget from a specific editor in the Qwt designer plugin.

Warning

The plot editor has never been implemented.

12.65.4.2 bool QwtPlot::autoReplot () const

Returns

true if the autoReplot option is set.

See Also

setAutoReplot()

12.65.4.3 bool QwtPlot::axisAutoScale (int axisId) const

Returns

True, if autoscaling is enabled

axisId Axis index

12.65.4.4 bool QwtPlot::axisEnabled (int axisId) const

Returns

True, if a specified axis is enabled

Parameters

axisId Axis index

12.65.4.5 QFont QwtPlot::axisFont (int axisId) const

Returns

The font of the scale labels for a specified axis

Parameters

axisId Axis index

12.65.4.6 QwtInterval QwtPlot::axisInterval (int axisId) const

Return the current interval of the specified axis.

This is only a convenience function for axisScaleDiv(axisId)->interval();

Parameters

axisId Axis index

Returns

Scale interval

See Also

QwtScaleDiv, axisScaleDiv()

12.65.4.7 int QwtPlot::axisMaxMajor (int axisId) const

Returns

The maximum number of major ticks for a specified axis

Parameters

axisId Axis index

See Also

setAxisMaxMajor(), QwtScaleEngine::divideScale()

12.65.4.8 int QwtPlot::axisMaxMinor (int axisId) const

Returns

the maximum number of minor ticks for a specified axis

Parameters

axisId Axis index

See Also

setAxisMaxMinor(), QwtScaleEngine::divideScale()

12.65.4.9 const QwtScaleDiv & QwtPlot::axisScaleDiv (int axisId) const

Return the scale division of a specified axis.

axisScaleDiv(axisId).lowerBound(), axisScaleDiv(axisId).upperBound() are the current limits of the axis scale.

Parameters

axisId Axis index

Returns

Scale division

See Also

QwtScaleDiv, setAxisScaleDiv(), QwtScaleEngine::divideScale()

12.65.4.10 const QwtScaleDraw * QwtPlot::axisScaleDraw (int axisId) const

Return the scale draw of a specified axis.

Parameters

axisId Axis index

Returns

Specified scaleDraw for axis, or NULL if axis is invalid.

12.65.4.11 QwtScaleDraw * QwtPlot::axisScaleDraw (int axisId)

Return the scale draw of a specified axis.

Parameters

axisId Axis index

Returns

Specified scaleDraw for axis, or NULL if axis is invalid.

12.65.4.12 QwtScaleEngine * QwtPlot::axisScaleEngine (int axisId)

Parameters

axisId Axis index

Returns

Scale engine for a specific axis

12.65.4.13 const QwtScaleEngine * QwtPlot::axisScaleEngine (int axisId) const

axisId Axis index

Returns

Scale engine for a specific axis

12.65.4.14 double QwtPlot::axisStepSize (int axisId) const

Return the step size parameter that has been set in setAxisScale.

This doesn't need to be the step size of the current scale.

Parameters

axisId Axis index

Returns

step size parameter value

See Also

setAxisScale(), QwtScaleEngine::divideScale()

12.65.4.15 QwtText QwtPlot::axisTitle (int axisId) const

Returns

Title of a specified axis

Parameters

axisId Axis index

12.65.4.16 bool QwtPlot::axisValid (int axisId) [static], [protected]

Returns

true if the specified axis exists, otherwise false

Parameters

axisId axis index

12.65.4.17 const QwtScaleWidget * QwtPlot::axisWidget (int axisId) const

Returns

Scale widget of the specified axis, or NULL if axisId is invalid.

Parameters

axisId Axis index

Returns

Scale widget of the specified axis, or NULL if axisId is invalid.

Parameters

axisId Axis index

12.65.4.19 QWidget * QwtPlot::canvas ()

Returns

the plot's canvas

12.65.4.20 const QWidget * QwtPlot::canvas () const

Returns

the plot's canvas

12.65.4.21 QBrush QwtPlot::canvasBackground () const

Nothing else than: canvas()->palette().brush(QPalette::Normal, QPalette::Window);

Returns

Background brush of the plotting area.

See Also

setCanvasBackground()

12.65.4.22 QwtScaleMap QwtPlot::canvasMap (int axisId) const [virtual]

Parameters

axisId Axis

Returns

Map for the axis on the canvas. With this map pixel coordinates can translated to plot coordinates and vice versa.

See Also

QwtScaleMap, transform(), invTransform()

12.65.4.23 void QwtPlot::drawCanvas (QPainter * painter) [virtual]

Redraw the canvas.

Parameters

painter | Painter used for drawing

Warning

drawCanvas calls drawItems what is also used for printing. Applications that like to add individual plot items better overload drawItems()

See Also

drawltems()

12.65.4.24 void QwtPlot::drawItems (QPainter * painter, const QRectF & canvasRect, const QwtScaleMap maps[axisCnt]) const [virtual]

Redraw the canvas items.

Parameters

| painte | Painter used for drawing |
|-----------|--|
| canvasRed | Bounding rectangle where to paint |
| map | QwtPlot::axisCnt maps, mapping between plot and paint device coordinates |

Note

Usually canvasRect is contentsRect() of the plot canvas. Due to a bug in Qt this rectangle might be wrong for certain frame styles (f.e QFrame::Box) and it might be necessary to fix the margins manually using QWidget::setContentsMargins()

12.65.4.25 void QwtPlot::enableAxis (int axisId, bool tf = true)

Enable or disable a specified axis.

When an axis is disabled, this only means that it is not visible on the screen. Curves, markers and can be attached to disabled axes, and transformation of screen coordinates into values works as normal.

Only xBottom and yLeft are enabled by default.

Parameters

| axisld | Axis index |
|--------|------------------------------------|
| tf | true (enabled) or false (disabled) |

12.65.4.26 bool QwtPlot::event (QEvent * event) [virtual]

Adds handling of layout requests.

Parameters

| event | Event |
|-------|-------|
|-------|-------|

Returns

See QFrame::event()

12.65.4.27 bool QwtPlot::eventFilter (QObject * object, QEvent * event) [virtual]

Event filter.

The plot handles the following events for the canvas:

- · QEvent::Resize The canvas margins might depend on its size
- QEvent::ContentsRectChange The layout needs to be recalculated

Parameters

| object | Object to be filtered |
|--------|-----------------------|
| event | Event |

Returns

See QFrame::eventFilter()

See Also

updateCanvasMargins(), updateLayout()

```
12.65.4.28 QwtText QwtPlot::footer ( ) const
```

Returns

Text of the footer

12.65.4.29 QwtTextLabel * QwtPlot::footerLabel ()

Returns

Footer label widget.

12.65.4.30 const QwtTextLabel * QwtPlot::footerLabel () const

Returns

Footer label widget.

12.65.4.31 void QwtPlot::getCanvasMarginsHint (const QwtScaleMap maps[], const QRectF & canvasRect, double & left, double & top, double & right, double & bottom) const [virtual]

Calculate the canvas margins.

Parameters

| maps | QwtPlot::axisCnt maps, mapping between plot and paint device coordinates |
|------------|--|
| canvasRect | Bounding rectangle where to paint |
| left | Return parameter for the left margin |
| top | Return parameter for the top margin |
| right | Return parameter for the right margin |
| bottom | Return parameter for the bottom margin |

Plot items might indicate, that they need some extra space at the borders of the canvas by the QwtPlotItem::Margins flag.

updateCanvasMargins(), QwtPlotItem::getCanvasMarginHint()

12.65.4.32 QString QwtPlot::grabProperties () const

This method is intended for manipulating the plot widget from a specific editor in the Qwt designer plugin.

Returns

QString::null

Warning

The plot editor has never been implemented.

12.65.4.33 QwtPlotItem * QwtPlot::infoToItem (const QVariant & itemInfo) const [virtual]

Identify the plot item according to an item info object, that has bee generated from itemToInfo().

The default implementation simply tries to unwrap a QwtPlotItem pointer:

```
if ( itemInfo.canConvert<QwtPlotItem *>() )
    return qvariant_cast<QwtPlotItem *>( itemInfo );
```

Parameters

| itemInfo | Plot item |
|----------|-----------|
|----------|-----------|

Returns

A plot item, when successful, otherwise a NULL pointer.

See Also

itemToInfo()

12.65.4.34 void QwtPlot::insertLegend (QwtAbstractLegend * legend, QwtPlot::LegendPosition pos = QwtPlot::RightLegend, double ratio = -1. 0)

Insert a legend.

If the position legend is <code>QwtPlot::LeftLegend</code> or <code>QwtPlot::RightLegend</code> the legend will be organized in one column from top to down. Otherwise the legend items will be placed in a table with a best fit number of columns from left to right.

insertLegend() will set the plot widget as parent for the legend. The legend will be deleted in the destructor of the plot or when another legend is inserted.

Legends, that are not inserted into the layout of the plot widget need to connect to the legendDataChanged() signal. Calling updateLegend() initiates this signal for an initial update. When the application code wants to implement its own layout this also needs to be done for rendering plots to a document (see QwtPlotRenderer).

Parameters

| legend | Legend |
|--------|---|
| pos | The legend's position. For top/left position the number of columns will be limited to 1, other- |
| | wise it will be set to unlimited. |
| ratio | Ratio between legend and the bounding rectangle of title, canvas and axes. The legend will |
| | be shrunk if it would need more space than the given ratio. The ratio is limited to]0.0 1.0]. |
| | In case of <= 0.0 it will be reset to the default ratio. The default vertical/horizontal ratio is |
| | 0.33/0.5. |

See Also

legend(), QwtPlotLayout::legendPosition(), QwtPlotLayout::setLegendPosition()

12.65.4.35 double QwtPlot::invTransform (int axisId, int pos) const

Transform the x or y coordinate of a position in the drawing region into a value.

Parameters

| axisId | Axis index |
|--------|------------|
| pos | position |

Returns

Position as axis coordinate

Warning

The position can be an x or a y coordinate, depending on the specified axis.

12.65.4.36 void QwtPlot::itemAttached (QwtPlotItem * plotItem, bool on) [signal]

A signal indicating, that an item has been attached/detached

| plotItem | Plot item |
|----------|-------------------|
| on | Attached/Detached |

12.65.4.37 QVariant QwtPlot::itemToInfo (QwtPlotItem * plotItem) const [virtual]

Build an information, that can be used to identify a plot item on the legend.

The default implementation simply wraps the plot item into a QVariant object. When overloading itemToInfo() usually infoToItem() needs to reimplemented too.

```
QVariant itemInfo;
qVariantSetValue( itemInfo, plotItem );
```

Parameters

| plotItem | Plot item |
|----------|-----------|

Returns

Plot item embedded in a QVariant

See Also

infoToItem()

12.65.4.38 QwtAbstractLegend * QwtPlot::legend ()

Returns

the plot's legend

See Also

insertLegend()

12.65.4.39 const QwtAbstractLegend * QwtPlot::legend () const

Returns

the plot's legend

See Also

insertLegend()

12.65.4.40 void QwtPlot::legendDataChanged (const QVariant & *itemInfo*, const QList< QwtLegendData > & *data*) [signal]

A signal with the attributes how to update the legend entries for a plot item.

Parameters

| itemInfo | Info about a plot item, build from itemToInfo() |
|----------|---|

data Attributes of the entries (usually <= 1) for the plot item.

See Also

itemToInfo(), infoToItem(), QwtAbstractLegend::updateLegend()

12.65.4.41 QwtPlotLayout * QwtPlot::plotLayout ()

Returns

the plot's layout

12.65.4.42 const QwtPlotLayout * QwtPlot::plotLayout () const

Returns

the plot's layout

12.65.4.43 void QwtPlot::replot() [virtual],[slot]

Redraw the plot.

If the autoReplot option is not set (which is the default) or if any curves are attached to raw data, the plot has to be refreshed explicitly in order to make changes visible.

See Also

updateAxes(), setAutoReplot()

12.65.4.44 void QwtPlot::resizeEvent (QResizeEvent * *e* **)** [protected], [virtual]

Resize and update internal layout

Parameters

e Resize event

12.65.4.45 void QwtPlot::setAutoReplot (bool tf = true)

Set or reset the autoReplot option.

If the autoReplot option is set, the plot will be updated implicitly by manipulating member functions. Since this may be time-consuming, it is recommended to leave this option switched off and call replot() explicitly if necessary.

The autoReplot option is set to false by default, which means that the user has to call replot() in order to make changes visible.

Parameters

tf | true or false. Defaults to true.

See Also

replot()

12.65.4.46 void QwtPlot::setAxisAutoScale (int axisId, bool on = true)

Enable autoscaling for a specified axis.

This member function is used to switch back to autoscaling mode after a fixed scale has been set. Autoscaling is enabled by default.

| axisld | Axis index |
|--------|------------|
| on | On/Off |

See Also

setAxisScale(), setAxisScaleDiv(), updateAxes()

Note

The autoscaling flag has no effect until updateAxes() is executed (called by replot()).

12.65.4.47 void QwtPlot::setAxisFont (int axisId, const QFont & font)

Change the font of an axis.

Parameters

| ſ | axisld | Axis index |
|---|--------|------------|
| | font | Font |

Warning

This function changes the font of the tick labels, not of the axis title.

12.65.4.48 void QwtPlot::setAxisLabelAlignment (int axisId, Qt::Alignment alignment)

Change the alignment of the tick labels

Parameters

| | axisld | Axis index |
|---|-----------|---|
| ſ | alignment | Or'd Qt::AlignmentFlags see <qnamespace.h></qnamespace.h> |

See Also

QwtScaleDraw::setLabelAlignment()

12.65.4.49 void QwtPlot::setAxisLabelRotation (int axisId, double rotation)

Rotate all tick labels

Parameters

| axisId | Axis index |
|----------|---|
| rotation | Angle in degrees. When changing the label rotation, the label alignment might be adjusted |
| | too. |

See Also

QwtScaleDraw::setLabelRotation(), setAxisLabelAlignment()

12.65.4.50 void QwtPlot::setAxisMaxMajor (int axisId, int maxMajor)

Set the maximum number of major scale intervals for a specified axis

Parameters

| axisld | Axis index |
|----------|-------------------------------|
| maxMajor | Maximum number of major steps |

See Also

axisMaxMajor()

12.65.4.51 void QwtPlot::setAxisMaxMinor (int axisId, int maxMinor)

Set the maximum number of minor scale intervals for a specified axis

Parameters

| axisId | Axis index |
|----------|-------------------------------|
| maxMinor | Maximum number of minor steps |

See Also

axisMaxMinor()

12.65.4.52 void QwtPlot::setAxisScale (int axisId, double min, double max, double stepSize = 0)

Disable autoscaling and specify a fixed scale for a selected axis.

In updateAxes() the scale engine calculates a scale division from the specified parameters, that will be assigned to the scale widget. So updates of the scale widget usually happen delayed with the next replot.

Parameters

| axisld | Axis index |
|----------|--|
| min | Minimum of the scale |
| max | Maximum of the scale |
| stepSize | Major step size. If step == 0, the step size is calculated automatically using the maxMajor setting. |

See Also

setAxisMaxMajor(), setAxisAutoScale(), axisStepSize(), QwtScaleEngine::divideScale()

12.65.4.53 void QwtPlot::setAxisScaleDiv (int axisId, const QwtScaleDiv & scaleDiv)

Disable autoscaling and specify a fixed scale for a selected axis.

The scale division will be stored locally only until the next call of updateAxes(). So updates of the scale widget usually happen delayed with the next replot.

Parameters

| axisId | Axis index |
|----------|----------------|
| scaleDiv | Scale division |

See Also

setAxisScale(), setAxisAutoScale()

12.65.4.54 void QwtPlot::setAxisScaleDraw (int axisId, QwtScaleDraw * scaleDraw)

Set a scale draw.

| axisld | Axis index |
|-----------|--|
| scaleDraw | Object responsible for drawing scales. |

By passing scaleDraw it is possible to extend QwtScaleDraw functionality and let it take place in QwtPlot. Please note that scaleDraw has to be created with new and will be deleted by the corresponding QwtScale member (like a child object).

See Also

QwtScaleDraw, QwtScaleWidget

Warning

The attributes of scaleDraw will be overwritten by those of the previous QwtScaleDraw.

12.65.4.55 void QwtPlot::setAxisScaleEngine (int axisId, QwtScaleEngine * scaleEngine)

Change the scale engine for an axis

Parameters

| axisld | Axis index |
|-------------|--------------|
| scaleEngine | Scale engine |

See Also

axisScaleEngine()

12.65.4.56 void QwtPlot::setAxisTitle (int axisId, const QString & title)

Change the title of a specified axis.

Parameters

| axisld | Axis index |
|--------|------------|
| title | axis title |

12.65.4.57 void QwtPlot::setAxisTitle (int axisId, const QwtText & title)

Change the title of a specified axis.

Parameters

| axisld | Axis index |
|--------|------------|
| title | Axis title |

12.65.4.58 void QwtPlot::setCanvas (QWidget * canvas)

Set the drawing canvas of the plot widget.

QwtPlot invokes methods of the canvas as meta methods (see QMetaObject). In opposite to using conventional C++ techniques like virtual methods they allow to use canvas implementations that are derived from QWidget or QGLWidget.

The following meta methods could be implemented:

- replot() When the canvas doesn't offer a replot method, QwtPlot calls update() instead.
- borderPath() The border path is necessary to clip the content of the canvas When the canvas doesn't have any special border (f.e rounded corners) it is o.k. not to implement this method.

The default canvas is a QwtPlotCanvas

Parameters

canvas Canvas Widget

See Also

canvas()

12.65.4.59 void QwtPlot::setCanvasBackground (const QBrush & brush)

Change the background of the plotting area.

Sets brush to QPalette::Window of all color groups of the palette of the canvas. Using canvas()->setPalette() is a more powerful way to set these colors.

Parameters

brush New background brush

See Also

canvasBackground()

12.65.4.60 void QwtPlot::setFooter (const QString & text)

Change the text the footer

Parameters

text New text of the footer

12.65.4.61 void QwtPlot::setFooter (const QwtText & text)

Change the text the footer

Parameters

text | New text of the footer

12.65.4.62 void QwtPlot::setPlotLayout (QwtPlotLayout * layout)

Assign a new plot layout.

Parameters

layout Layout()

See Also

plotLayout()

12.65.4.63 void QwtPlot::setTitle (const QString & title)

Change the plot's title

Parameters

title New title

12.65.4.64 void QwtPlot::setTitle (const QwtText & title)

Change the plot's title

Parameters

title New title

12.65.4.65 QSize QwtPlot::sizeHint() const [virtual]

Returns

Size hint for the plot widget

See Also

minimumSizeHint()

12.65.4.66 QwtText QwtPlot::title () const

Returns

Title of the plot

12.65.4.67 QwtTextLabel * QwtPlot::titleLabel ()

Returns

Title label widget.

12.65.4.68 const QwtTextLabel * QwtPlot::titleLabel () const

Returns

Title label widget.

12.65.4.69 double QwtPlot::transform (int axisId, double value) const

Transform a value into a coordinate in the plotting region.

Parameters

| axisld | Axis index |
|--------|------------|
| value | value |

Returns

X or Y coordinate in the plotting region corresponding to the value.

12.65.4.70 void QwtPlot::updateAxes ()

Rebuild the axes scales.

In case of autoscaling the boundaries of a scale are calculated from the bounding rectangles of all plot items, having the QwtPlotItem::AutoScale flag enabled (QwtScaleEngine::autoScale()). Then a scale division is calculated (QwtScaleEngine::didvideScale()) and assigned to scale widget.

When the scale boundaries have been assigned with setAxisScale() a scale division is calculated (QwtScale-Engine::didvideScale()) for this interval and assigned to the scale widget.

When the scale has been set explicitly by setAxisScaleDiv() the locally stored scale division gets assigned to the scale widget.

The scale widget indicates modifications by emitting a QwtScaleWidget::scaleDivChanged() signal. updateAxes() is usually called by replot().

```
See Also
```

setAxisAutoScale(), setAxisScale(), setAxisScaleDiv(), replot() QwtPlotItem::boundingRect()

```
12.65.4.71 void QwtPlot::updateCanvasMargins ( )
```

Update the canvas margins.

Plot items might indicate, that they need some extra space at the borders of the canvas by the QwtPlotItem::Margins flag.

getCanvasMarginsHint(), QwtPlotItem::getCanvasMarginHint()

```
12.65.4.72 void QwtPlot::updateLayout( ) [virtual]
```

Adjust plot content to its current size.

See Also

resizeEvent()

12.65.4.73 void QwtPlot::updateLegend ()

Emit legendDataChanged() for all plot item

See Also

QwtPlotItem::legendData(), legendDataChanged()

12.65.4.74 void QwtPlot::updateLegend (const QwtPlotItem * plotItem)

Emit legendDataChanged() for a plot item

Parameters

plotItem Plot item

See Also

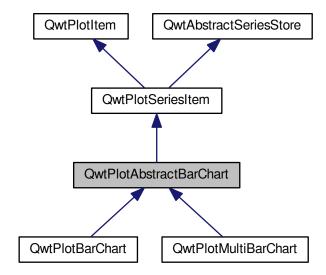
QwtPlotItem::legendData(), legendDataChanged()

12.66 QwtPlotAbstractBarChart Class Reference

Abstract base class for bar chart items.

#include <qwt_plot_abstract_barchart.h>

Inheritance diagram for QwtPlotAbstractBarChart:



Public Types

• enum LayoutPolicy { AutoAdjustSamples, ScaleSamplesToAxes, ScaleSampleToCanvas, FixedSampleSize } Mode how to calculate the bar width.

Public Member Functions

- QwtPlotAbstractBarChart (const QwtText &title)
- virtual ~QwtPlotAbstractBarChart ()

Destructor.

- void setLayoutPolicy (LayoutPolicy)
- LayoutPolicy layoutPolicy () const
- void setLayoutHint (double)
- double layoutHint () const
- void setSpacing (int)

Set the spacing.

- int spacing () const
- void setMargin (int)

Set the margin.

• int margin () const

• void setBaseline (double)

Set the baseline.

- · double baseline () const
- virtual void getCanvasMarginHint (const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, double &left, double &top, double &right, double &bottom) const

Calculate a hint for the canvas margin.

Protected Member Functions

double sampleWidth (const QwtScaleMap &map, double canvasSize, double dataSize, double value) const

12.66.1 Detailed Description

Abstract base class for bar chart items.

In opposite to almost all other plot items bar charts can't be displayed inside of their bounding rectangle and need a special API how to calculate the width of the bars and how they affect the layout of the attached plot.

12.66.2 Member Enumeration Documentation

12.66.2.1 enum QwtPlotAbstractBarChart::LayoutPolicy

Mode how to calculate the bar width.

setLayoutPolicy(), setLayoutHint(), barWidthHint()

Enumerator

AutoAdjustSamples The sample width is calculated by dividing the bounding rectangle by the number of samples. The layoutHint() is used as a minimum width in paint device coordinates.

See Also

boundingRectangle()

ScaleSamplesToAxes layoutHint() defines an interval in axis coordinates

ScaleSampleToCanvas The bar width is calculated by multiplying layoutHint() with the height or width of the canvas.

See Also

boundingRectangle()

FixedSampleSize layoutHint() defines a fixed width in paint device coordinates.

12.66.3 Constructor & Destructor Documentation

12.66.3.1 QwtPlotAbstractBarChart::QwtPlotAbstractBarChart (const QwtText & title) [explicit]

Constructor

Parameters

| title | Title of the chart |
|-------|--------------------|

12.66.4 Member Function Documentation

12.66.4.1 double QwtPlotAbstractBarChart::baseline () const

Returns

Value for the origin of the bar chart

See Also

setBaseline(), QwtPlotSeriesItem::orientation()

12.66.4.2 void QwtPlotAbstractBarChart::getCanvasMarginHint (const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, double & left, double & top, double & right, double & bottom) const [virtual]

Calculate a hint for the canvas margin.

Bar charts need to reserve some space for displaying the bars for the first and the last sample. The hint is calculated from the layoutHint() depending on the layoutPolicy().

The margins are in target device coordinates (pixels on screen)

Parameters

| хМар | Maps x-values into pixel coordinates. |
|------------|---|
| уМар | Maps y-values into pixel coordinates. |
| canvasRect | Contents rectangle of the canvas in painter coordinates |
| left | Returns the left margin |
| top | Returns the top margin |
| right | Returns the right margin |
| bottom | Returns the bottom margin |

Returns

Margin

See Also

layoutPolicy(), layoutHint(), QwtPlotItem::Margins QwtPlot::getCanvasMarginsHint(), QwtPlot::updateCanvas-Margins()

Reimplemented from QwtPlotItem.

12.66.4.3 double QwtPlotAbstractBarChart::layoutHint () const

The combination of layoutPolicy() and layoutHint() define how the width of the bars is calculated

Returns

Layout policy of the chart item

See Also

LayoutPolicy, setLayoutHint(), layoutPolicy()

12.66.4.4 QwtPlotAbstractBarChart::LayoutPolicy QwtPlotAbstractBarChart::layoutPolicy () const

The combination of layoutPolicy() and layoutHint() define how the width of the bars is calculated

Returns

Layout policy of the chart item

See Also

setLayoutPolicy(), layoutHint()

12.66.4.5 int QwtPlotAbstractBarChart::margin () const

Returns

Margin between the outmost bars and the contentsRect() of the canvas.

See Also

setMargin(), spacing()

12.66.4.6 double QwtPlotAbstractBarChart::sampleWidth (const QwtScaleMap & map, double canvasSize, double boundingSize, double value) const [protected]

Calculate the width for a sample in paint device coordinates

Parameters

| тар | Scale map for the corresponding scale |
|--------------|---|
| canvasSize | Size of the canvas in paint device coordinates |
| boundingSize | Bounding size of the chart in plot coordinates (used in AutoAdjustSamples mode) |
| value | Value of the sample |

Returns

Sample width

See Also

layoutPolicy(), layoutHint()

12.66.4.7 void QwtPlotAbstractBarChart::setBaseline (double value)

Set the baseline.

The baseline is the origin for the chart. Each bar is painted from the baseline in the direction of the sample value. In case of a horizontal orientation() the baseline is interpreted as x - otherwise as y - value.

The default value for the baseline is 0.

Parameters

|--|

See Also

baseline(), QwtPlotSeriesItem::orientation()

12.66.4.8 void QwtPlotAbstractBarChart::setLayoutHint (double hint)

The combination of layoutPolicy() and layoutHint() define how the width of the bars is calculated

Parameters

| hint | Layout hint |
|------|-------------|

See Also

LayoutPolicy, layoutPolicy(), layoutHint()

12.66.4.9 void QwtPlotAbstractBarChart::setLayoutPolicy (LayoutPolicy policy)

The combination of layoutPolicy() and layoutHint() define how the width of the bars is calculated

| policy | Layout policy |
|--------|---------------|
|--------|---------------|

See Also

layoutPolicy(), layoutHint()

12.66.4.10 void QwtPlotAbstractBarChart::setMargin (int margin)

Set the margin.

The margin is the distance between the outmost bars and the contentsRect() of the canvas. The default setting is 5 pixels.

Parameters

```
margin Margin
```

See Also

spacing(), margin()

12.66.4.11 void QwtPlotAbstractBarChart::setSpacing (int spacing)

Set the spacing.

The spacing is the distance between 2 samples (bars for QwtPlotBarChart or a group of bars for QwtPlotMultiBarChart) in paint device coordinates.

See Also

spacing()

12.66.4.12 int QwtPlotAbstractBarChart::spacing () const

Returns

Spacing between 2 samples (bars or groups of bars)

See Also

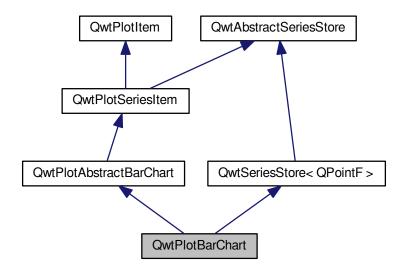
setSpacing(), margin()

12.67 QwtPlotBarChart Class Reference

QwtPlotBarChart displays a series of a values as bars.

```
#include <qwt_plot_barchart.h>
```

Inheritance diagram for QwtPlotBarChart:



Public Types

enum LegendMode { LegendChartTitle, LegendBarTitles }
 Legend modes.

Public Member Functions

- QwtPlotBarChart (const QString &title=QString::null)
- QwtPlotBarChart (const QwtText &title)
- virtual ~QwtPlotBarChart ()

Destructor.

- · virtual int rtti () const
- void setSamples (const QVector< QPointF > &)
- void setSamples (const QVector< double > &)
- void setSamples (QwtSeriesData < QPointF > *series)
- void setSymbol (QwtColumnSymbol *)

Assign a symbol.

- const QwtColumnSymbol * symbol () const
- void setLegendMode (LegendMode)
- LegendMode legendMode () const

- virtual void drawSeries (QPainter *painter, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int from, int to) const
- · virtual QRectF boundingRect () const
- virtual QwtColumnSymbol * specialSymbol (int sampleIndex, const QPointF &) const
- virtual QwtText barTitle (int sampleIndex) const

Return the title of a bar.

Protected Member Functions

- virtual void drawSample (QPainter *painter, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, const QwtInterval &boundingInterval, int index, const QPointF &sample) const
- virtual void drawBar (QPainter *, int sampleIndex, const QPointF &point, const QwtColumnRect &) const
- QList< QwtLegendData > legendData () const

Return all information, that is needed to represent the item on the legend.

QwtGraphic legendlcon (int index, const QSizeF &) const

12.67.1 Detailed Description

QwtPlotBarChart displays a series of a values as bars.

Each bar might be customized individually by implementing a special Symbol(). Otherwise it is rendered using a default symbol.

Depending on its orientation() the bars are displayed horizontally or vertically. The bars cover the interval between the baseline() and the value.

By activating the LegendBarTitles mode each sample will have its own entry on the legend.

The most common use case of a bar chart is to display a list of y coordinates, where the x coordinate is simply the index in the list. But for other situations (f.e. when values are related to dates) it is also possible to set x coordinates explicitly.

See Also

QwtPlotMultiBarChart, QwtPlotHistogram, QwtPlotCurve::Sticks, QwtPlotSeriesItem::orientation(), QwtPlotAbstractBarChart::baseline()

12.67.2 Member Enumeration Documentation

12.67.2.1 enum QwtPlotBarChart::LegendMode

Legend modes.

The default setting is QwtPlotBarChart::LegendChartTitle.

See Also

setLegendMode(), legendMode()

Enumerator

LegendChartTitle One entry on the legend showing the default symbol and the title() of the chart See Also

QwtPlotItem::title()

LegendBarTitles One entry for each value showing the individual symbol of the corresponding bar and the bar title.

See Also

specialSymbol(), barTitle()

12.67.3 Constructor & Destructor Documentation

12.67.3.1 QwtPlotBarChart::QwtPlotBarChart (const QString & title = QString::null) [explicit]

Constructor

Parameters

title Title of the curve

12.67.3.2 QwtPlotBarChart::QwtPlotBarChart (const QwtText & title) [explicit]

Constructor

Parameters

title Title of the curve

12.67.4 Member Function Documentation

12.67.4.1 QwtText QwtPlotBarChart::barTitle (int sampleIndex) const [virtual]

Return the title of a bar.

In LegendBarTitles mode the title is displayed on the legend entry corresponding to a bar.

The default implementation is a dummy, that is intended to be overloaded.

Parameters

sampleIndex Index of the bar

Returns

An empty text

See Also

LegendBarTitles

12.67.4.2 QRectF QwtPlotBarChart::boundingRect() const [virtual]

Returns

Bounding rectangle of all samples. For an empty series the rectangle is invalid.

Reimplemented from QwtPlotSeriesItem.

12.67.4.3 void QwtPlotBarChart::drawBar (QPainter * painter, int sampleIndex, const QPointF & sample, const QwtColumnRect & rect) const [protected], [virtual]

Draw a bar

Parameters

| painter | Painter |
|-------------|--|
| sampleIndex | Index of the sample represented by the bar |

| sample | Value of the sample |
|--------|-------------------------------|
| rect | Bounding rectangle of the bar |

12.67.4.4 void QwtPlotBarChart::drawSample (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, const QwtInterval & boundingInterval, int index, const QPointF & sample) const [protected], [virtual]

Draw a sample

Parameters

| painter | Painter |
|------------------|------------------------------------|
| хМар | х тар |
| уМар | y map |
| canvasRect | Contents rect of the canvas |
| boundingInterval | Bounding interval of sample values |
| index | Index of the sample |
| sample | Value of the sample |

See Also

drawSeries()

12.67.4.5 void QwtPlotBarChart::drawSeries (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [virtual]

Draw an interval of the bar chart

Parameters

| painter | Painter |
|------------|---|
| хМар | Maps x-values into pixel coordinates. |
| уМар | Maps y-values into pixel coordinates. |
| canvasRect | Contents rect of the canvas |
| from | Index of the first point to be painted |
| to | Index of the last point to be painted. If to $<$ 0 the curve will be painted to its last point. |

See Also

drawSymbols()

Implements QwtPlotSeriesItem.

12.67.4.6 QList< QwtLegendData > QwtPlotBarChart::legendData () const [protected], [virtual]

Return all information, that is needed to represent the item on the legend.

In case of LegendBarTitles an entry for each bar is returned, otherwise the chart is represented like any other plot item from its title() and the legendlcon().

Returns

Information, that is needed to represent the item on the legend

See Also

title(), setLegendMode(), barTitle(), QwtLegend, QwtPlotLegendItem

Reimplemented from QwtPlotItem.

12.67.4.7 QwtGraphic QwtPlotBarChart::legendlcon (int *index***, const QSizeF &** *size* **) const** [protected], [virtual]

Returns

Icon representing a bar or the chart on the legend

When the legendMode() is LegendBarTitles the icon shows the bar corresponding to index - otherwise the bar displays the default symbol.

Parameters

| index | Index of the legend entry |
|-------|---------------------------|
| size | Icon size |

See Also

setLegendMode(), drawBar(), QwtPlotItem::setLegendIconSize(), QwtPlotItem::legendData()

Reimplemented from QwtPlotItem.

12.67.4.8 QwtPlotBarChart::LegendMode QwtPlotBarChart::legendMode () const

Returns

Legend mode

See Also

setLegendMode()

12.67.4.9 int QwtPlotBarChart::rtti() const [virtual]

Returns

QwtPlotItem::Rtti_PlotBarChart

Reimplemented from QwtPlotItem.

12.67.4.10 void QwtPlotBarChart::setLegendMode (LegendMode mode)

Set the mode that decides what to display on the legend

In case of LegendBarTitles barTitle() needs to be overloaded to return individual titles for each bar.

Parameters

| mode | New mode |
|------|----------|

See Also

legendMode(), legendData(), barTitle(), QwtPlotItem::ItemAttribute

12.67.4.11 void QwtPlotBarChart::setSamples (const QVector< QPointF > & samples)

Initialize data with an array of points

samples Vector of points

Note

QVector is implicitly shared QPolygonF is derived from QVector<QPointF>

12.67.4.12 void QwtPlotBarChart::setSamples (const QVector< double > & samples)

Initialize data with an array of doubles

The indices in the array are taken as x coordinate, while the doubles are interpreted as y values.

Parameters

samples | Vector of y coordinates

Note

QVector is implicitly shared

12.67.4.13 void QwtPlotBarChart::setSamples (QwtSeriesData < QPointF > * data)

Assign a series of samples

setSamples() is just a wrapper for setData() without any additional value - beside that it is easier to find for the developer.

Parameters

| 33.00 |
|-------|
|-------|

Warning

The item takes ownership of the data object, deleting it when its not used anymore.

12.67.4.14 void QwtPlotBarChart::setSymbol (QwtColumnSymbol * symbol)

Assign a symbol.

The bar chart will take the ownership of the symbol, hence the previously set symbol will be delete by setting a new one. If symbol is NULL no symbol will be drawn.

Parameters

symbol Symbol

See Also

symbol()

12.67.4.15 QwtColumnSymbol * QwtPlotBarChart::specialSymbol (int sampleIndex, const QPointF & sample) const [virtual]

Needs to be overloaded to return a non default symbol for a specific sample

Parameters

| sampleIndex | Index of the sample represented by the bar |
|-------------|--|
| sample | Value of the sample |

Returns

NULL, indicating to use the default symbol

12.67.4.16 const QwtColumnSymbol * QwtPlotBarChart::symbol () const

Returns

Current symbol or NULL, when no symbol has been assigned

See Also

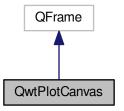
setSymbol()

12.68 QwtPlotCanvas Class Reference

Canvas of a QwtPlot.

#include <qwt_plot_canvas.h>

Inheritance diagram for QwtPlotCanvas:



Public Types

- enum PaintAttribute { BackingStore = 1, Opaque = 2, HackStyledBackground = 4, ImmediatePaint = 8 }

 Paint attributes.
- enum FocusIndicator { NoFocusIndicator, CanvasFocusIndicator, ItemFocusIndicator }

Focus indicator The default setting is NoFocusIndicator.

typedef QFlags < PaintAttribute > PaintAttributes
 Paint attributes.

Public Slots

void replot ()

Public Member Functions

QwtPlotCanvas (QwtPlot *=NULL)

Constructor.

virtual ~QwtPlotCanvas ()

Destructor.

QwtPlot * plot ()

Return parent plot widget.

const QwtPlot * plot () const

Return parent plot widget.

- · void setFocusIndicator (FocusIndicator)
- FocusIndicator focusIndicator () const
- void setBorderRadius (double)
- · double borderRadius () const
- void setPaintAttribute (PaintAttribute, bool on=true)

Changing the paint attributes.

- bool testPaintAttribute (PaintAttribute) const
- const QPixmap * backingStore () const
- void invalidateBackingStore ()

Invalidate the internal backing store.

- virtual bool event (QEvent *)
- Q_INVOKABLE QPainterPath borderPath (const QRect &) const

Protected Member Functions

- virtual void paintEvent (QPaintEvent *)
- virtual void resizeEvent (QResizeEvent *)
- virtual void drawFocusIndicator (QPainter *)
- virtual void drawBorder (QPainter *)
- void updateStyleSheetInfo ()

Update the cached information about the current style sheet.

12.68.1 Detailed Description

Canvas of a QwtPlot.

Canvas is the widget where all plot items are displayed

See Also

QwtPlot::setCanvas(), QwtPlotGLCanvas

12.68.2 Member Enumeration Documentation

12.68.2.1 enum QwtPlotCanvas::FocusIndicator

Focus indicator The default setting is NoFocusIndicator.

See Also

setFocusIndicator(), focusIndicator(), paintFocus()

Enumerator

NoFocusIndicator Don't paint a focus indicator.

CanvasFocusIndicator The focus is related to the complete canvas. Paint the focus indicator using paint-Focus()

ItemFocusIndicator The focus is related to an item (curve, point, ...) on the canvas. It is up to the application to display a focus indication using f.e. highlighting.

12.68.2.2 enum QwtPlotCanvas::PaintAttribute

Paint attributes.

The default setting enables BackingStore and Opaque.

See Also

setPaintAttribute(), testPaintAttribute()

Enumerator

BackingStore Paint double buffered reusing the content of the pixmap buffer when possible. Using a backing store might improve the performance significantly, when working with widget overlays (like rubber bands). Disabling the cache might improve the performance for incremental paints (using QwtPlotDirectPainter).

See Also

backingStore(), invalidateBackingStore()

Opaque Try to fill the complete contents rectangle of the plot canvas. When using styled backgrounds Qt assumes, that the canvas doesn't fill its area completely (f.e because of rounded borders) and fills the area below the canvas. When this is done with gradients it might result in a serious performance bottleneck - depending on the size.

When the Opaque attribute is enabled the canvas tries to identify the gaps with some heuristics and to fill those only.

Warning

Will not work for semitransparent backgrounds

HackStyledBackground Try to improve painting of styled backgrounds. QwtPlotCanvas supports the box model attributes for customizing the layout with style sheets. Unfortunately the design of Qt style sheets has no concept how to handle backgrounds with rounded corners - beside of padding.

When HackStyledBackground is enabled the plot canvas tries to separate the background from the background border by reverse engineering to paint the background before and the border after the plot items. In this order the border gets perfectly antialiased and you can avoid some pixel artifacts in the corners.

ImmediatePaint When ImmediatePaint is set replot() calls repaint() instead of update().

See Also

```
replot(), QWidget::repaint(), QWidget::update()
```

12.68.3 Constructor & Destructor Documentation

12.68.3.1 QwtPlotCanvas::QwtPlotCanvas (QwtPlot * plot = NULL) [explicit]

Constructor.

plot Parent plot widget

See Also

QwtPlot::setCanvas()

12.68.4 Member Function Documentation

12.68.4.1 const QPixmap * QwtPlotCanvas::backingStore () const

Returns

Backing store, might be null

Calculate the painter path for a styled or rounded border

When the canvas has no styled background or rounded borders the painter path is empty.

Parameters

rect Bounding rectangle of the canvas

Returns

Painter path, that can be used for clipping

12.68.4.3 double QwtPlotCanvas::borderRadius () const

Returns

Radius for the corners of the border frame

See Also

setBorderRadius()

12.68.4.4 void QwtPlotCanvas::drawBorder (QPainter * painter) [protected], [virtual]

Draw the border of the plot canvas

Parameters

painter Painter

See Also

setBorderRadius()

12.68.4.5 void QwtPlotCanvas::drawFocusIndicator(QPainter * painter) [protected], [virtual]

Draw the focus indication

Parameters

painter Painter

12.68.4.6 bool QwtPlotCanvas::event (QEvent * event) [virtual]

Qt event handler for QEvent::PolishRequest and QEvent::StyleChange

Parameters

event | Qt Event

Returns

See QFrame::event()

12.68.4.7 QwtPlotCanvas::FocusIndicator QwtPlotCanvas::focusIndicator () const

Returns

Focus indicator

See Also

FocusIndicator, setFocusIndicator()

12.68.4.8 void QwtPlotCanvas::paintEvent (QPaintEvent * event) [protected], [virtual]

Paint event

Parameters

event Paint event

12.68.4.9 void QwtPlotCanvas::replot() [slot]

Invalidate the paint cache and repaint the canvas

See Also

invalidatePaintCache()

12.68.4.10 void QwtPlotCanvas::resizeEvent (QResizeEvent * event) [protected], [virtual]

Resize event

Parameters

event Resize event

12.68.4.11 void QwtPlotCanvas::setBorderRadius (double radius)

Set the radius for the corners of the border frame

Parameters

| radius | Radius of a rounded corner |
|--------|----------------------------|
| | riadido di a rodindo dorno |

See Also

borderRadius()

12.68.4.12 void QwtPlotCanvas::setFocusIndicator (FocusIndicator focusIndicator)

Set the focus indicator

See Also

FocusIndicator, focusIndicator()

12.68.4.13 void QwtPlotCanvas::setPaintAttribute (PaintAttribute attribute, bool on = true)

Changing the paint attributes.

Parameters

| attribute | Paint attribute |
|-----------|-----------------|
| on | On/Off |

See Also

testPaintAttribute(), backingStore()

 $12.68.4.14 \quad bool \ QwtPlotCanvas:: testPaintAttribute \ (\ PaintAttribute \ attribute \) \ const$

Test whether a paint attribute is enabled

Parameters

| attribute | Paint attribute |
|-----------|-----------------|
|-----------|-----------------|

Returns

true, when attribute is enabled

See Also

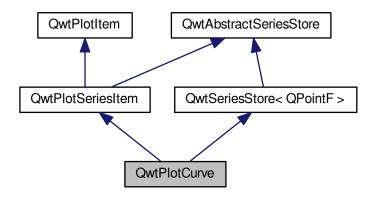
setPaintAttribute()

12.69 QwtPlotCurve Class Reference

A plot item, that represents a series of points.

```
#include <qwt_plot_curve.h>
```

Inheritance diagram for QwtPlotCurve:



Public Types

- enum CurveStyle {
 NoCurve = -1, Lines, Sticks, Steps,
 Dots, UserCurve = 100 }
- enum CurveAttribute { Inverted = 0x01, Fitted = 0x02 }
- enum LegendAttribute { LegendNoAttribute = 0x00, LegendShowLine = 0x01, LegendShowSymbol = 0x02, LegendShowBrush = 0x04 }
- enum PaintAttribute { ClipPolygons = 0x01, FilterPoints = 0x02, MinimizeMemory = 0x04, ImageBuffer = 0x08 }
- typedef QFlags < CurveAttribute > CurveAttributes

Curve attributes.

typedef QFlags < LegendAttribute > LegendAttributes

Legend attributes.

typedef QFlags< PaintAttribute > PaintAttributes

Paint attributes.

Public Member Functions

- QwtPlotCurve (const QString &title=QString::null)
- QwtPlotCurve (const QwtText &title)
- virtual ~QwtPlotCurve ()

Destructor.

- virtual int rtti () const
- void setPaintAttribute (PaintAttribute, bool on=true)

- bool testPaintAttribute (PaintAttribute) const
- void setLegendAttribute (LegendAttribute, bool on=true)
- bool testLegendAttribute (LegendAttribute) const
- void setRawSamples (const double *xData, const double *yData, int size)

Initialize the data by pointing to memory blocks which are not managed by QwtPlotCurve.

- void setSamples (const double *xData, const double *yData, int size)
- void setSamples (const QVector< double > &xData, const QVector< double > &yData)

Initialize data with x- and y-arrays (explicitly shared)

- void setSamples (const QVector< QPointF > &)
- void setSamples (QwtSeriesData < QPointF > *)
- int closestPoint (const QPoint &pos, double *dist=NULL) const
- double minXValue () const

boundingRect().left()

• double maxXValue () const

boundingRect().right()

• double minYValue () const

boundingRect().top()

· double maxYValue () const

boundingRect().bottom()

- void setCurveAttribute (CurveAttribute, bool on=true)
- bool testCurveAttribute (CurveAttribute) const
- void setPen (const QColor &, greal width=0.0, Qt::PenStyle=Qt::SolidLine)
- void setPen (const QPen &)
- const QPen & pen () const
- void setBrush (const QBrush &)

Assign a brush.

- · const QBrush & brush () const
- void setBaseline (double)

Set the value of the baseline.

- double baseline () const
- void setStyle (CurveStyle style)
- CurveStyle style () const
- void setSymbol (QwtSymbol *)

Assign a symbol.

- const QwtSymbol * symbol () const
- void setCurveFitter (QwtCurveFitter *)
- QwtCurveFitter * curveFitter () const
- virtual void drawSeries (QPainter *, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int from, int to) const
- · virtual QwtGraphic legendlcon (int index, const QSizeF &) const

Protected Member Functions

• void init ()

Initialize internal members.

 virtual void drawCurve (QPainter *p, int style, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int from, int to) const

Draw the line part (without symbols) of a curve interval.

- virtual void drawSymbols (QPainter *p, const QwtSymbol &, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int from, int to) const
- virtual void drawLines (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int from, int to) const

Draw lines.

 virtual void drawSticks (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int from, int to) const

- virtual void drawDots (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int from, int to) const
- virtual void drawSteps (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int from, int to) const
- virtual void fillCurve (QPainter *, const QwtScaleMap &, const QwtScaleMap &, const QRectF &canvasRect, QPolygonF &) const
- void closePolyline (QPainter *, const QwtScaleMap &, const QwtScaleMap &, QPolygonF &) const
 Complete a polygon to be a closed polygon including the area between the original polygon and the baseline.

12.69.1 Detailed Description

A plot item, that represents a series of points.

A curve is the representation of a series of points in the x-y plane. It supports different display styles, interpolation (f.e. spline) and symbols.

Usage

- a) Assign curve properties When a curve is created, it is configured to draw black solid lines with in QwtPlot-Curve::Lines style and no symbols. You can change this by calling setPen(), setStyle() and setSymbol().
- b) Connect/Assign data. QwtPlotCurve gets its points using a QwtSeriesData object offering a bridge to the real storage of the points (like QAbstractItemModel). There are several convenience classes derived from QwtSeriesData, that also store the points inside (like QStandardItemModel). QwtPlotCurve also offers a couple of variations of setSamples(), that build QwtSeriesData objects from arrays internally.
- c) Attach the curve to a plot See QwtPlotItem::attach()

Example:

see examples/bode

See Also

QwtPointSeriesData, QwtSymbol, QwtScaleMap

12.69.2 Member Enumeration Documentation

12.69.2.1 enum QwtPlotCurve::CurveAttribute

Attribute for drawing the curve

See Also

setCurveAttribute(), testCurveAttribute(), curveFitter()

Enumerator

Inverted For QwtPlotCurve::Steps only. Draws a step function from the right to the left.

Fitted Only in combination with QwtPlotCurve::Lines A QwtCurveFitter tries to interpolate/smooth the curve, before it is painted.

Note

Curve fitting requires temporary memory for calculating coefficients and additional points. If painting in QwtPlotCurve::Fitted mode is slow it might be better to fit the points, before they are passed to QwtPlotCurve.

12.69.2.2 enum QwtPlotCurve::CurveStyle

Curve styles.

See Also

setStyle(), style()

Enumerator

NoCurve Don't draw a curve. Note: This doesn't affect the symbols.

Lines Connect the points with straight lines. The lines might be interpolated depending on the 'Fitted' attribute. Curve fitting can be configured using setCurveFitter().

Sticks Draw vertical or horizontal sticks (depending on the orientation()) from a baseline which is defined by setBaseline().

Steps Connect the points with a step function. The step function is drawn from the left to the right or vice versa, depending on the QwtPlotCurve::Inverted attribute.

Dots Draw dots at the locations of the data points. Note: This is different from a dotted line (see setPen()), and faster as a curve in QwtPlotCurve::NoStyle style and a symbol painting a point.

UserCurve Styles >= QwtPlotCurve::UserCurve are reserved for derived classes of QwtPlotCurve that overload drawCurve() with additional application specific curve types.

12.69.2.3 enum QwtPlotCurve::LegendAttribute

Attributes how to represent the curve on the legend

See Also

setLegendAttribute(), testLegendAttribute(), QwtPlotItem::legendData(), legendIcon()

Enumerator

LegendNoAttribute QwtPlotCurve tries to find a color representing the curve and paints a rectangle with it.

LegendShowLine If the style() is not QwtPlotCurve::NoCurve a line is painted with the curve pen().

LegendShowSymbol If the curve has a valid symbol it is painted.

LegendShowBrush If the curve has a brush a rectangle filled with the curve brush() is painted.

12.69.2.4 enum QwtPlotCurve::PaintAttribute

Attributes to modify the drawing algorithm. The default setting enables ClipPolygons | FilterPoints

See Also

setPaintAttribute(), testPaintAttribute()

Enumerator

ClipPolygons Clip polygons before painting them. In situations, where points are far outside the visible area (f.e when zooming deep) this might be a substantial improvement for the painting performance

FilterPoints Tries to reduce the data that has to be painted, by sorting out duplicates, or paintings outside the visible area. Might have a notable impact on curves with many close points. Only a couple of very basic filtering algorithms are implemented.

MinimizeMemory Minimize memory usage that is temporarily needed for the translated points, before they get painted. This might slow down the performance of painting

ImageBuffer Render the points to a temporary image and paint the image. This is a very special optimization for Dots style, when having a huge amount of points. With a reasonable number of points QPainter::draw-Points() will be faster.

12.69.3 Constructor & Destructor Documentation

12.69.3.1 QwtPlotCurve::QwtPlotCurve(const QString & title = QString::null) [explicit]

Constructor

| title | Title of the curve |
|-------|--------------------|
|-------|--------------------|

12.69.3.2 QwtPlotCurve::QwtPlotCurve(const QwtText & title) [explicit]

Constructor

Parameters

| title | Title of the curve |
|-------|--------------------|

12.69.4 Member Function Documentation

12.69.4.1 double QwtPlotCurve::baseline () const

Returns

Value of the baseline

See Also

setBaseline()

12.69.4.2 const QBrush & QwtPlotCurve::brush () const

Returns

Brush used to fill the area between lines and the baseline

See Also

setBrush(), setBaseline(), baseline()

12.69.4.3 void QwtPlotCurve::closePolyline (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, QPolygonF & polygon) const [protected]

Complete a polygon to be a closed polygon including the area between the original polygon and the baseline.

Parameters

| painter | Painter |
|---------|-------------------------|
| хМар | X map |
| уМар | Y map |
| polygon | Polygon to be completed |

12.69.4.4 int QwtPlotCurve::closestPoint (const QPoint & pos, double * dist = NULL) const

Find the closest curve point for a specific position

Parameters

| pos | Position, where to look for the closest curve point |
|------|---|
| dist | If dist != NULL, closestPoint() returns the distance between the position and the closest curve |
| | point |

Returns

Index of the closest curve point, or -1 if none can be found (f.e when the curve has no points)

Note

closestPoint() implements a dumb algorithm, that iterates over all points

12.69.4.5 QwtCurveFitter * QwtPlotCurve::curveFitter () const

Get the curve fitter. If curve fitting is disabled NULL is returned.

Returns

Curve fitter

See Also

setCurveFitter(), Fitted

12.69.4.6 void QwtPlotCurve::drawCurve (QPainter * painter, int style, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [protected], [virtual]

Draw the line part (without symbols) of a curve interval.

Parameters

| painter | Painter |
|------------|---|
| style | curve style, see QwtPlotCurve::CurveStyle |
| хМар | x map |
| уМар | y map |
| canvasRect | Contents rectangle of the canvas |
| from | index of the first point to be painted |
| to | index of the last point to be painted |

See Also

draw(), drawDots(), drawLines(), drawSteps(), drawSticks()

12.69.4.7 void QwtPlotCurve::drawDots (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [protected], [virtual]

Draw dots

Parameters

| painter | Painter |
|------------|--|
| хМар | x map |
| уМар | y map |
| canvasRect | Contents rectangle of the canvas |
| from | index of the first point to be painted |
| to | index of the last point to be painted |

See Also

draw(), drawCurve(), drawSticks(), drawLines(), drawSteps()

12.69.4.8 void QwtPlotCurve::drawLines (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [protected], [virtual]

Draw lines.

If the CurveAttribute Fitted is enabled a QwtCurveFitter tries to interpolate/smooth the curve, before it is painted.

| painter | Painter |
|------------|--|
| хМар | x map |
| уМар | y map |
| canvasRect | Contents rectangle of the canvas |
| from | index of the first point to be painted |
| to | index of the last point to be painted |

See Also

setCurveAttribute(), setCurveFitter(), draw(), drawLines(), drawDots(), drawSteps(), drawSticks()

12.69.4.9 void QwtPlotCurve::drawSeries (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [virtual]

Draw an interval of the curve

Parameters

| painter | Painter |
|------------|---|
| хМар | Maps x-values into pixel coordinates. |
| уМар | Maps y-values into pixel coordinates. |
| canvasRect | Contents rectangle of the canvas |
| from | Index of the first point to be painted |
| to | Index of the last point to be painted. If to $<$ 0 the curve will be painted to its last point. |

See Also

drawCurve(), drawSymbols(),

Implements QwtPlotSeriesItem.

12.69.4.10 void QwtPlotCurve::drawSteps (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [protected], [virtual]

Draw step function

The direction of the steps depends on Inverted attribute.

Parameters

| painter | Painter |
|------------|--|
| хМар | х тар |
| уМар | y map |
| canvasRect | Contents rectangle of the canvas |
| from | index of the first point to be painted |
| to | index of the last point to be painted |

See Also

CurveAttribute, setCurveAttribute(), draw(), drawCurve(), drawDots(), drawLines(), drawSticks()

12.69.4.11 void QwtPlotCurve::drawSticks (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [protected], [virtual]

Draw sticks

Parameters

| painter | Painter |
|------------|--|
| хМар | x map |
| уМар | y map |
| canvasRect | Contents rectangle of the canvas |
| from | index of the first point to be painted |
| to | index of the last point to be painted |

See Also

draw(), drawCurve(), drawDots(), drawLines(), drawSteps()

12.69.4.12 void QwtPlotCurve::drawSymbols (QPainter * painter, const QwtSymbol & symbol, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [protected], [virtual]

Draw symbols

Parameters

| painter | Painter |
|------------|--|
| symbol | Curve symbol |
| хМар | x map |
| уМар | y map |
| canvasRect | Contents rectangle of the canvas |
| from | Index of the first point to be painted |
| to | Index of the last point to be painted |

See Also

setSymbol(), drawSeries(), drawCurve()

12.69.4.13 void QwtPlotCurve::fillCurve (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, QPolygonF & polygon) const [protected], [virtual]

Fill the area between the curve and the baseline with the curve brush

Parameters

| painter | Painter |
|------------|----------------------------------|
| хМар | x map |
| уМар | y map |
| canvasRect | Contents rectangle of the canvas |
| polygon | Polygon - will be modified ! |

See Also

setBrush(), setBaseline(), setStyle()

12.69.4.14 QwtGraphic QwtPlotCurve::legendlcon (int index, const QSizeF & size) const [virtual]

Returns

Icon representing the curve on the legend

Parameters

| index | Index of the legend entry (ignored as there is only one) |
|-------|--|
| size | Icon size |

See Also

QwtPlotItem::setLegendIconSize(), QwtPlotItem::legendData()

Reimplemented from QwtPlotItem.

12.69.4.15 const QPen & QwtPlotCurve::pen () const

Returns

Pen used to draw the lines

See Also

setPen(), brush()

12.69.4.16 int QwtPlotCurve::rtti() const [virtual]

Returns

QwtPlotItem::Rtti_PlotCurve

Reimplemented from QwtPlotItem.

12.69.4.17 void QwtPlotCurve::setBaseline (double value)

Set the value of the baseline.

The baseline is needed for filling the curve with a brush or the Sticks drawing style.

The interpretation of the baseline depends on the orientation(). With Qt::Horizontal, the baseline is interpreted as a horizontal line at y = baseline(), with Qt::Vertical, it is interpreted as a vertical line at x = baseline().

The default value is 0.0.

Parameters

| value | Value of the baseline |
|-------|-----------------------|
|-------|-----------------------|

See Also

baseline(), setBrush(), setStyle(), QwtPlotAbstractSeriesItem::orientation()

12.69.4.18 void QwtPlotCurve::setBrush (const QBrush & brush)

Assign a brush.

In case of brush.style() != QBrush::NoBrush and style() != QwtPlotCurve::Sticks the area between the curve and the baseline will be filled.

In case !brush.color().isValid() the area will be filled by pen.color(). The fill algorithm simply connects the first and the last curve point to the baseline. So the curve data has to be sorted (ascending or descending).

Parameters

| brush | New brush |
|-------|-----------|
|-------|-----------|

See Also

brush(), setBaseline(), baseline()

12.69.4.19 void QwtPlotCurve::setCurveAttribute (CurveAttribute attribute, bool on = true)

Specify an attribute for drawing the curve

Parameters

| attribute | Curve attribute |
|-----------|-----------------|
| on | On/Off |

/sa testCurveAttribute(), setCurveFitter()

12.69.4.20 void QwtPlotCurve::setCurveFitter (QwtCurveFitter * curveFitter)

Assign a curve fitter

The curve fitter "smooths" the curve points, when the Fitted CurveAttribute is set. setCurveFitter(NULL) also disables curve fitting.

The curve fitter operates on the translated points (= widget coordinates) to be functional for logarithmic scales. Obviously this is less performant for fitting algorithms, that reduce the number of points.

For situations, where curve fitting is used to improve the performance of painting huge series of points it might be better to execute the fitter on the curve points once and to cache the result in the QwtSeriesData object.

Parameters

| curveFitter() | Curve fitter |
|---------------|--------------|
|---------------|--------------|

See Also

Fitted

12.69.4.21 void QwtPlotCurve::setLegendAttribute (LegendAttribute attribute, bool on = true)

Specify an attribute how to draw the legend icon

Parameters

| attribute | Attribute |
|-----------|--|
| on | On/Off /sa testLegendAttribute(). legendIcon() |

12.69.4.22 void QwtPlotCurve::setPaintAttribute (PaintAttribute attribute, bool on = true)

Specify an attribute how to draw the curve

Parameters

| attribute | Paint attribute |
|-----------|-----------------|
| on | On/Off |

See Also

testPaintAttribute()

12.69.4.23 void QwtPlotCurve::setPen (const QColor & color, qreal width = 0 . 0, Qt::PenStyle style = Qt::SolidLine)

Build and assign a pen

In Qt5 the default pen width is 1.0 (0.0 in Qt4) what makes it non cosmetic (see QPen::isCosmetic()). This method has been introduced to hide this incompatibility.

Parameters

| color | Pen color |
|-------|-----------|
| width | Pen width |
| style | Pen style |

See Also

pen(), brush()

12.69.4.24 void QwtPlotCurve::setPen (const QPen & pen)

Assign a pen

Parameters

| pen | New pen |
|-----|---------|
|-----|---------|

See Also

pen(), brush()

12.69.4.25 void QwtPlotCurve::setRawSamples (const double * xData, const double * yData, int size)

Initialize the data by pointing to memory blocks which are not managed by QwtPlotCurve.

setRawSamples is provided for efficiency. It is important to keep the pointers during the lifetime of the underlying QwtCPointerData class.

Parameters

| xData | pointer to x data |
|-------|-------------------|
| yData | pointer to y data |
| size | size of x and y |

See Also

QwtCPointerData

12.69.4.26 void QwtPlotCurve::setSamples (const double * xData, const double * yData, int size)

Set data by copying x- and y-values from specified memory blocks. Contrary to setRawSamples(), this function makes a 'deep copy' of the data.

Parameters

| xData | pointer to x values |
|-------|-------------------------|
| yData | pointer to y values |
| size | size of xData and yData |

See Also

QwtPointArrayData

12.69.4.27 void QwtPlotCurve::setSamples (const QVector< double > & xData, const QVector< double > & yData)

Initialize data with x- and y-arrays (explicitly shared)

Parameters

| xData | x data |
|-------|--------|
| yData | y data |

See Also

QwtPointArrayData

12.69.4.28 void QwtPlotCurve::setSamples (const QVector < QPointF > & samples)

Initialize data with an array of points.

Parameters

| samples | Vector of points |
|---------|------------------|

Note

QVector is implicitly shared

QPolygonF is derived from QVector<QPointF>

12.69.4.29 void QwtPlotCurve::setSamples (QwtSeriesData < QPointF > * data)

Assign a series of points

setSamples() is just a wrapper for setData() without any additional value - beside that it is easier to find for the developer.

Parameters

| data | Data |
|------|------|
|------|------|

Warning

The item takes ownership of the data object, deleting it when its not used anymore.

12.69.4.30 void QwtPlotCurve::setStyle (CurveStyle style)

Set the curve's drawing style

Parameters

| style | Curve style |
|-------|-------------|

See Also

style()

12.69.4.31 void QwtPlotCurve::setSymbol (QwtSymbol * symbol)

Assign a symbol.

The curve will take the ownership of the symbol, hence the previously set symbol will be delete by setting a new one. If symbol is NULL no symbol will be drawn.

```
Parameters
```

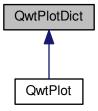
```
symbol
                      Symbol
See Also
      symbol()
12.69.4.32 QwtPlotCurve::CurveStyle QwtPlotCurve::style ( ) const
Returns
      Style of the curve
See Also
      setStyle()
12.69.4.33 const QwtSymbol * QwtPlotCurve::symbol ( ) const
Returns
      Current symbol or NULL, when no symbol has been assigned
See Also
      setSymbol()
12.69.4.34 bool QwtPlotCurve::testCurveAttribute ( CurveAttribute attribute ) const
Returns
      true, if attribute is enabled
See Also
      setCurveAttribute()
12.69.4.35 bool QwtPlotCurve::testLegendAttribute ( LegendAttribute attribute ) const
Returns
      True, when attribute is enabled
See Also
      setLegendAttribute()
12.69.4.36 bool QwtPlotCurve::testPaintAttribute ( PaintAttribute attribute ) const
Returns
      True, when attribute is enabled
See Also
      setPaintAttribute()
```

12.70 QwtPlotDict Class Reference

A dictionary for plot items.

#include <qwt_plot_dict.h>

Inheritance diagram for QwtPlotDict:



Public Member Functions

- QwtPlotDict ()
- virtual ~QwtPlotDict ()
- void setAutoDelete (bool)
- bool autoDelete () const
- const QwtPlotItemList & itemList () const

A QwtPlotItemList of all attached plot items.

- QwtPlotItemList itemList (int rtti) const
- void detachItems (int rtti=QwtPlotItem::Rtti_PlotItem, bool autoDelete=true)

Protected Member Functions

- void insertItem (QwtPlotItem *)
- void removeltem (QwtPlotItem *)

12.70.1 Detailed Description

A dictionary for plot items.

QwtPlotDict organizes plot items in increasing z-order. If autoDelete() is enabled, all attached items will be deleted in the destructor of the dictionary. QwtPlotDict can be used to get access to all QwtPlotItem items - or all items of a specific type - that are currently on the plot.

See Also

QwtPlotItem::attach(), QwtPlotItem::detach(), QwtPlotItem::z()

12.70.2 Constructor & Destructor Documentation

12.70.2.1 QwtPlotDict::QwtPlotDict() [explicit]

Constructor

Auto deletion is enabled.

```
See Also
```

setAutoDelete(), QwtPlotItem::attach()

12.70.2.2 QwtPlotDict::~QwtPlotDict() [virtual]

Destructor

If autoDelete() is on, all attached items will be deleted

See Also

setAutoDelete(), autoDelete(), QwtPlotItem::attach()

12.70.3 Member Function Documentation

12.70.3.1 bool QwtPlotDict::autoDelete () const

Returns

true if auto deletion is enabled

See Also

setAutoDelete(), insertItem()

12.70.3.2 void QwtPlotDict::detachItems (int rtti = QwtPlotItem::Rtti_PlotItem, bool autoDelete = true)

Detach items from the dictionary

Parameters

| rtti | In case of QwtPlotItem::Rtti_PlotItem detach all items otherwise only those items of the type |
|------------|---|
| | rtti. |
| autoDelete | If true, delete all detached items |

12.70.3.3 void QwtPlotDict::insertItem (QwtPlotItem * item) [protected]

Insert a plot item

Parameters

| item | PlotItem |
|------|----------|

See Also

removeItem()

12.70.3.4 const QwtPlotItemList & QwtPlotDict::itemList () const

A QwtPlotItemList of all attached plot items.

Use caution when iterating these lists, as removing/detaching an item will invalidate the iterator. Instead you can place pointers to objects to be removed in a removal list, and traverse that list later.

Returns

List of all attached plot items.

12.70.3.5 QwtPlotItemList QwtPlotDict::itemList (int rtti) const

Returns

List of all attached plot items of a specific type.

Parameters

| rtti | See QwtPlotItem::RttiValues |
|------|-----------------------------|
| | |

See Also

QwtPlotItem::rtti()

12.70.3.6 void QwtPlotDict::removeItem (QwtPlotItem * item) [protected]

Remove a plot item

Parameters

| item | PlotItem |
|------|----------|
|------|----------|

See Also

insertItem()

12.70.3.7 void QwtPlotDict::setAutoDelete (bool autoDelete)

En/Disable Auto deletion

If Auto deletion is on all attached plot items will be deleted in the destructor of QwtPlotDict. The default value is on.

See Also

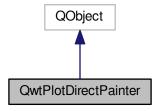
autoDelete(), insertItem()

12.71 QwtPlotDirectPainter Class Reference

Painter object trying to paint incrementally.

#include <qwt_plot_directpainter.h>

Inheritance diagram for QwtPlotDirectPainter:



Public Types

enum Attribute { AtomicPainter = 0x01, FullRepaint = 0x02, CopyBackingStore = 0x04 }

Paint attributes.

typedef QFlags< Attribute > Attributes

Paint attributes.

Public Member Functions

QwtPlotDirectPainter (QObject *parent=NULL)

Constructor.

virtual ~QwtPlotDirectPainter ()

Destructor.

- · void setAttribute (Attribute, bool on)
- bool testAttribute (Attribute) const
- void setClipping (bool)
- · bool hasClipping () const
- void setClipRegion (const QRegion &)

Assign a clip region and enable clipping.

- QRegion clipRegion () const
- void drawSeries (QwtPlotSeriesItem *, int from, int to)

Draw a set of points of a seriesItem.

· void reset ()

Close the internal QPainter.

virtual bool eventFilter (QObject *, QEvent *)

Event filter.

12.71.1 Detailed Description

Painter object trying to paint incrementally.

Often applications want to display samples while they are collected. When there are too many samples complete replots will be expensive to be processed in a collection cycle.

QwtPlotDirectPainter offers an API to paint subsets (f.e all additions points) without erasing/repainting the plot canvas.

On certain environments it might be important to calculate a proper clip region before painting. F.e. for Qt Embedded only the clipped part of the backing store will be copied to a (maybe unaccelerated) frame buffer.

Warning

Incremental painting will only help when no replot is triggered by another operation (like changing scales) and nothing needs to be erased.

12.71.2 Member Enumeration Documentation

12.71.2.1 enum QwtPlotDirectPainter::Attribute

Paint attributes.

See Also

setAttribute(), testAttribute(), drawSeries()

Enumerator

AtomicPainter Initializing a QPainter is an expensive operation. When AtomicPainter is set each call of drawSeries() opens/closes a temporary QPainter. Otherwise QwtPlotDirectPainter tries to use the same QPainter as long as possible.

FullRepaint When FullRepaint is set the plot canvas is explicitly repainted after the samples have been rendered.

CopyBackingStore When QwtPlotCanvas::BackingStore is enabled the painter has to paint to the backing store and the widget. In certain situations/environments it might be faster to paint to the backing store only and then copy the backing store to the canvas. This flag can also be useful for settings, where Qt fills the the clip region with the widget background.

12.71.3 Member Function Documentation

12.71.3.1 QRegion QwtPlotDirectPainter::clipRegion () const

Returns

Currently set clip region.

See Also

setClipRegion(), setClipping(), hasClipping()

12.71.3.2 void QwtPlotDirectPainter::drawSeries (QwtPlotSeriesItem * seriesItem, int from, int to)

Draw a set of points of a seriesItem.

When observing an measurement while it is running, new points have to be added to an existing seriesItem. draw-Series() can be used to display them avoiding a complete redraw of the canvas.

Setting plot()->canvas()->setAttribute(Qt::WA_PaintOutsidePaintEvent, true); will result in faster painting, if the paint engine of the canvas widget supports this feature.

Parameters

| seriesItem | Item to be painted |
|------------|--|
| from | Index of the first point to be painted |
| to | Index of the last point to be painted. If to $<$ 0 the series will be painted to its last point. |

12.71.3.3 bool QwtPlotDirectPainter::hasClipping () const

Returns

true, when clipping is enabled

See Also

setClipping(), clipRegion(), setClipRegion()

12.71.3.4 void QwtPlotDirectPainter::setAttribute (Attribute attribute, bool on)

Change an attribute

Parameters

| attribute | Attribute to change |
|-----------|---------------------|
| on | On/Off |

See Also

Attribute, testAttribute()

12.71.3.5 void QwtPlotDirectPainter::setClipping (bool enable)

En/Disables clipping

Parameters

| enable | Enables clipping is true, disable it otherwise |
|--------|--|
|--------|--|

See Also

hasClipping(), clipRegion(), setClipRegion()

12.71.3.6 void QwtPlotDirectPainter::setClipRegion (const QRegion & region)

Assign a clip region and enable clipping.

Depending on the environment setting a proper clip region might improve the performance heavily. F.e. on Qt embedded only the clipped part of the backing store will be copied to a (maybe unaccelerated) frame buffer device.

Parameters

| region | Clip region |
|--------|-------------|
|--------|-------------|

See Also

clipRegion(), hasClipping(), setClipping()

12.71.3.7 bool QwtPlotDirectPainter::testAttribute (Attribute attribute) const

Returns

True, when attribute is enabled

Parameters

| attribute | Attribute to be tested |
|-----------|------------------------|

See Also

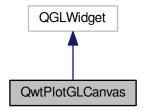
Attribute, setAttribute()

12.72 QwtPlotGLCanvas Class Reference

An alternative canvas for a QwtPlot derived from QGLWidget.

#include <qwt_plot_glcanvas.h>

Inheritance diagram for QwtPlotGLCanvas:



Public Types

- enum Shadow { Plain = QFrame::Plain, Raised = QFrame::Raised, Sunken = QFrame::Sunken }
 Frame shadow.
- enum Shape { NoFrame = QFrame::NoFrame, Box = QFrame::Box, Panel = QFrame::Panel }
 Frame shape.

Public Slots

• void replot ()

Calls repaint()

Public Member Functions

QwtPlotGLCanvas (QwtPlot *=NULL)

Constructor.

virtual ~QwtPlotGLCanvas ()

Destructor.

- void setFrameStyle (int style)
- int frameStyle () const
- · void setFrameShadow (Shadow)
- Shadow frameShadow () const
- void setFrameShape (Shape)
- Shape frameShape () const
- void setLineWidth (int)
- int lineWidth () const
- void setMidLineWidth (int)
- int midLineWidth () const
- int frameWidth () const
- QRect frameRect () const
- Q_INVOKABLE QPainterPath borderPath (const QRect &) const
- virtual bool event (QEvent *)

Protected Member Functions

- virtual void paintEvent (QPaintEvent *)
- virtual void drawBackground (QPainter *)
- virtual void drawBorder (QPainter *)
- virtual void drawltems (QPainter *)

12.72.1 Detailed Description

An alternative canvas for a QwtPlot derived from QGLWidget.

QwtPlotGLCanvas implements the very basics to act as canvas inside of a QwtPlot widget. It might be extended to a full featured alternative to QwtPlotCanvas in a future version of Qwt.

Even if QwtPlotGLCanvas is not derived from QFrame it imitates its API. When using style sheets it supports the box model - beside backgrounds with rounded borders.

See Also

QwtPlot::setCanvas(), QwtPlotCanvas

Note

You might want to use the QPaintEngine::OpenGL paint engine (see QGL::setPreferredPaintEngine()). On a Linux test system QPaintEngine::OpenGL2 shows very basic problems (wrong geometries of rectangles) but also more advanced stuff like antialiasing doesn't work.

Another way to introduce OpenGL rendering to Qwt is to use QGLPixelBuffer or QGLFramebufferObject. Both type of buffers can be converted into a QImage and used in combination with a regular QwtPlotCanvas.

12.72.2 Member Enumeration Documentation

12.72.2.1 enum QwtPlotGLCanvas::Shadow

Frame shadow.

Unfortunately it is not possible to use QFrame::Shadow as a property of a widget that is not derived from QFrame. The following enum is made for the designer only. It is safe to use QFrame::Shadow instead.

Enumerator

Plain QFrame::Plain.Raised QFrame::Raised.Sunken QFrame::Sunken.

12.72.2.2 enum QwtPlotGLCanvas::Shape

Frame shape.

Unfortunately it is not possible to use QFrame::Shape as a property of a widget that is not derived from QFrame. The following enum is made for the designer only. It is safe to use QFrame::Shadow instead.

Note

QFrame::StyledPanel and QFrame::WinPanel are unsuported and will be displayed as QFrame::Panel.

12.72.3 Constructor & Destructor Documentation

12.72.3.1 QwtPlotGLCanvas::QwtPlotGLCanvas (QwtPlot * plot = NULL) [explicit]

Constructor.

Parameters

plot Parent plot widget

See Also

QwtPlot::setCanvas()

12.72.4 Member Function Documentation

12.72.4.1 QPainterPath QwtPlotGLCanvas::borderPath (const QRect & rect) const

Returns

Empty path

12.72.4.2 void QwtPlotGLCanvas::drawBackground (QPainter * painter) [protected], [virtual]

Draw the background of the canvas

Parameters

painter Painter

12.72.4.3 void QwtPlotGLCanvas::drawBorder (QPainter * painter) [protected], [virtual]

Draw the border of the canvas

Parameters

painter Painter

12.72.4.4 void QwtPlotGLCanvas::drawItems (QPainter * painter) [protected], [virtual]

Draw the plot items

Parameters

painter Painter

See Also

QwtPlot::drawCanvas()

12.72.4.5 bool QwtPlotGLCanvas::event (QEvent * event) [virtual]

Qt event handler for QEvent::PolishRequest and QEvent::StyleChange

Parameters

event | Qt Event

Returns

See QGLWidget::event()

12.72.4.6 QRect QwtPlotGLCanvas::frameRect () const

Returns

The rectangle where the frame is drawn in.

```
12.72.4.7 QwtPlotGLCanvas::Shadow QwtPlotGLCanvas::frameShadow ( ) const
Returns
     Frame shadow
See Also
     setFrameShadow(), QFrame::setFrameShadow()
12.72.4.8 QwtPlotGLCanvas::Shape QwtPlotGLCanvas::frameShape ( ) const
Returns
     Frame shape
See Also
     setFrameShape(), QFrame::frameShape()
12.72.4.9 int QwtPlotGLCanvas::frameStyle ( ) const
Returns
     The bitwise OR between a frameShape() and a frameShadow()
See Also
     setFrameStyle(), QFrame::frameStyle()
12.72.4.10 int QwtPlotGLCanvas::frameWidth ( ) const
Returns
     Frame width depending on the style, line width and midline width.
12.72.4.11 int QwtPlotGLCanvas::lineWidth ( ) const
Returns
     Line width of the frame
See Also
     setLineWidth(), midLineWidth()
12.72.4.12 int QwtPlotGLCanvas::midLineWidth ( ) const
Returns
     Midline width of the frame
See Also
     setMidLineWidth(), lineWidth()
12.72.4.13 void QwtPlotGLCanvas::paintEvent ( QPaintEvent * event ) [protected], [virtual]
Paint event
```

Parameters

event Paint event

See Also

QwtPlot::drawCanvas()

12.72.4.14 void QwtPlotGLCanvas::setFrameShadow (Shadow shadow)

Set the frame shadow

Parameters

shadow Frame shadow

See Also

frameShadow(), setFrameShape(), QFrame::setFrameShadow()

12.72.4.15 void QwtPlotGLCanvas::setFrameShape (Shape shape)

Set the frame shape

Parameters

shape Frame shape

See Also

frameShape(), setFrameShadow(), QFrame::frameShape()

12.72.4.16 void QwtPlotGLCanvas::setFrameStyle (int style)

Set the frame style

Parameters

style The bitwise OR between a shape and a shadow.

See Also

 $frameStyle(),\ QFrame::setFrameStyle(),\ setFrameShadow(),\ setFrameShape()$

12.72.4.17 void QwtPlotGLCanvas::setLineWidth (int width)

Set the frame line width

The default line width is 2 pixels.

Parameters

width Line width of the frame

See Also

lineWidth(), setMidLineWidth()

12.72.4.18 void QwtPlotGLCanvas::setMidLineWidth (int width)

Set the frame mid line width

The default midline width is 0 pixels.

Parameters

| width | Midline width of the frame |
|-------|----------------------------|
|-------|----------------------------|

See Also

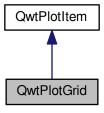
midLineWidth(), setLineWidth()

12.73 QwtPlotGrid Class Reference

A class which draws a coordinate grid.

#include <qwt_plot_grid.h>

Inheritance diagram for QwtPlotGrid:



Public Member Functions

• QwtPlotGrid ()

Enables major grid, disables minor grid.

virtual ~QwtPlotGrid ()

Destructor.

- · virtual int rtti () const
- void enableX (bool tf)

Enable or disable vertical grid lines.

- bool xEnabled () const
- void enableY (bool tf)

Enable or disable horizontal grid lines.

- bool yEnabled () const
- void enableXMin (bool tf)

Enable or disable minor vertical grid lines.

- bool xMinEnabled () const
- void enableYMin (bool tf)

Enable or disable minor horizontal grid lines.

- bool yMinEnabled () const
- void setXDiv (const QwtScaleDiv &sx)
- · const QwtScaleDiv & xScaleDiv () const
- void setYDiv (const QwtScaleDiv &sy)
- · const QwtScaleDiv & yScaleDiv () const
- void setPen (const QColor &, qreal width=0.0, Qt::PenStyle=Qt::SolidLine)
- void setPen (const QPen &)

- void setMajorPen (const QColor &, greal width=0.0, Qt::PenStyle=Qt::SolidLine)
- void setMajorPen (const QPen &)
- · const QPen & majorPen () const
- void setMinorPen (const QColor &, greal width=0.0, Qt::PenStyle=Qt::SolidLine)
- void setMinorPen (const QPen &p)
- const QPen & minorPen () const
- virtual void draw (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &rect) const

Draw the grid.

virtual void updateScaleDiv (const QwtScaleDiv &xMap, const QwtScaleDiv &yMap)

Additional Inherited Members

12.73.1 Detailed Description

A class which draws a coordinate grid.

The QwtPlotGrid class can be used to draw a coordinate grid. A coordinate grid consists of major and minor vertical and horizontal grid lines. The locations of the grid lines are determined by the X and Y scale divisions which can be assigned with setXDiv() and setYDiv(). The draw() member draws the grid within a bounding rectangle.

12.73.2 Member Function Documentation

12.73.2.1 void QwtPlotGrid::draw (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect) const [virtual]

Draw the grid.

The grid is drawn into the bounding rectangle such that grid lines begin and end at the rectangle's borders. The X and Y maps are used to map the scale divisions into the drawing region screen.

Parameters

| painter | Painter |
|------------|---------------------------------------|
| хМар | X axis map |
| уМар | Y axis |
| canvasRect | Contents rectangle of the plot canvas |

Implements QwtPlotItem.

12.73.2.2 void QwtPlotGrid::enableX (bool on)

Enable or disable vertical grid lines.

Parameters

| on Enable (true) or disable |
|-----------------------------|
|-----------------------------|

See Also

Minor grid lines can be enabled or disabled with enableXMin()

12.73.2.3 void QwtPlotGrid::enableXMin (bool on)

Enable or disable minor vertical grid lines.

```
Parameters
```

on Enable (true) or disable

See Also

enableX()

12.73.2.4 void QwtPlotGrid::enableY (bool on)

Enable or disable horizontal grid lines.

Parameters

on Enable (true) or disable

See Also

Minor grid lines can be enabled or disabled with enableYMin()

12.73.2.5 void QwtPlotGrid::enableYMin (bool on)

Enable or disable minor horizontal grid lines.

Parameters

on Enable (true) or disable

See Also

enableY()

12.73.2.6 const QPen & QwtPlotGrid::majorPen () const

Returns

the pen for the major grid lines

See Also

setMajorPen(), setMinorPen(), setPen()

12.73.2.7 const QPen & QwtPlotGrid::minorPen () const

Returns

the pen for the minor grid lines

See Also

setMinorPen(), setMajorPen(), setPen()

12.73.2.8 int QwtPlotGrid::rtti() const [virtual]

Returns

QwtPlotItem::Rtti_PlotGrid

Reimplemented from QwtPlotItem.

12.73.2.9 void QwtPlotGrid::setMajorPen (const QColor & color, qreal width = 0.0, Qt::PenStyle style = Qt::SolidLine)

Build and assign a pen for both major grid lines

In Qt5 the default pen width is 1.0 (0.0 in Qt4) what makes it non cosmetic (see QPen::isCosmetic()). This method has been introduced to hide this incompatibility.

Parameters

| color | Pen color |
|-------|-----------|
| width | Pen width |
| style | Pen style |

See Also

pen(), brush()

12.73.2.10 void QwtPlotGrid::setMajorPen (const QPen & pen)

Assign a pen for the major grid lines

Parameters

| pen | Pen |
|-----|-----|

See Also

majorPen(), setMinorPen(), setPen()

12.73.2.11 void QwtPlotGrid::setMinorPen (const QColor & color, qreal width = 0.0, Qt::PenStyle style = Qt::SolidLine)

Build and assign a pen for the minor grid lines

In Qt5 the default pen width is 1.0 (0.0 in Qt4) what makes it non cosmetic (see QPen::isCosmetic()). This method has been introduced to hide this incompatibility.

Parameters

| color | Pen color |
|-------|-----------|
| width | Pen width |
| style | Pen style |

See Also

pen(), brush()

12.73.2.12 void QwtPlotGrid::setMinorPen (const QPen & pen)

Assign a pen for the minor grid lines

Parameters

| pen | Pen |
|-----|-----|

See Also

minorPen(), setMajorPen(), setPen()

12.73.2.13 void QwtPlotGrid::setPen (const QColor & color, greal width = 0.0, Qt::PenStyle style = Qt::SolidLine)

Build and assign a pen for both major and minor grid lines

In Qt5 the default pen width is 1.0 (0.0 in Qt4) what makes it non cosmetic (see QPen::isCosmetic()). This method has been introduced to hide this incompatibility.

Parameters

| color | Pen color |
|-------|-----------|
| width | Pen width |
| style | Pen style |

See Also

pen(), brush()

12.73.2.14 void QwtPlotGrid::setPen (const QPen & pen)

Assign a pen for both major and minor grid lines

Parameters

| pen | Pen |
|-----|-----|
|-----|-----|

See Also

setMajorPen(), setMinorPen()

12.73.2.15 void QwtPlotGrid::setXDiv (const QwtScaleDiv & scaleDiv)

Assign an x axis scale division

Parameters

| scaleDiv | Scale division |
|----------|----------------|

12.73.2.16 void QwtPlotGrid::setYDiv (const QwtScaleDiv & scaleDiv)

Assign a y axis division

Parameters

| scaleDiv | Scale division |
|----------|----------------|
| Coarobii | |

12.73.2.17 void QwtPlotGrid::updateScaleDiv (const QwtScaleDiv & xScaleDiv, const QwtScaleDiv & yScaleDiv)
[virtual]

Update the grid to changes of the axes scale division

Parameters

| xScaleDiv | Scale division of the x-axis |
|-----------|------------------------------|
| yScaleDiv | Scale division of the y-axis |

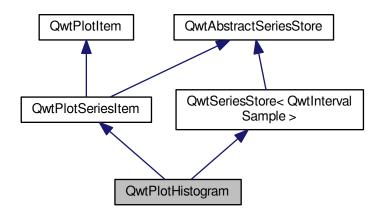
See Also

QwtPlot::updateAxes()

Reimplemented from QwtPlotItem.

```
12.73.2.18 bool QwtPlotGrid::xEnabled ( ) const
Returns
      true if vertical grid lines are enabled
See Also
      enableX()
12.73.2.19 bool QwtPlotGrid::xMinEnabled ( ) const
Returns
      true if minor vertical grid lines are enabled
See Also
      enableXMin()
12.73.2.20 const QwtScaleDiv & QwtPlotGrid::xScaleDiv ( ) const
Returns
      the scale division of the x axis
12.73.2.21 bool QwtPlotGrid::yEnabled ( ) const
Returns
      true if horizontal grid lines are enabled
See Also
      enableY()
12.73.2.22 bool QwtPlotGrid::yMinEnabled ( ) const
Returns
     true if minor horizontal grid lines are enabled
See Also
      enableYMin()
12.73.2.23 const QwtScaleDiv & QwtPlotGrid::yScaleDiv ( ) const
Returns
      the scale division of the y axis
12.74 QwtPlotHistogram Class Reference
QwtPlotHistogram represents a series of samples, where an interval is associated with a value (y = f([x1, x2])).
#include <qwt_plot_histogram.h>
```

Inheritance diagram for QwtPlotHistogram:



Public Types

enum HistogramStyle { Outline, Columns, Lines, UserStyle = 100 }

Public Member Functions

- QwtPlotHistogram (const QString &title=QString::null)
- QwtPlotHistogram (const QwtText &title)
- virtual ~QwtPlotHistogram ()

Destructor.

- virtual int rtti () const
- void setPen (const QColor &, qreal width=0.0, Qt::PenStyle=Qt::SolidLine)
- void setPen (const QPen &)
- · const QPen & pen () const
- void setBrush (const QBrush &)
- const QBrush & brush () const
- void setSamples (const QVector< QwtIntervalSample > &)
- void setSamples (QwtSeriesData< QwtIntervalSample > *)
- · void setBaseline (double reference)

Set the value of the baseline.

- · double baseline () const
- void setStyle (HistogramStyle style)
- HistogramStyle style () const
- void setSymbol (const QwtColumnSymbol *)

Assign a symbol.

- const QwtColumnSymbol * symbol () const
- virtual void drawSeries (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int from, int to) const
- virtual QRectF boundingRect () const
- virtual QwtGraphic legendlcon (int index, const QSizeF &) const

Protected Member Functions

virtual QwtColumnRect columnRect (const QwtIntervalSample &, const QwtScaleMap &, const QwtScaleMap &) const

- virtual void drawColumn (QPainter *, const QwtColumnRect &, const QwtIntervalSample &) const
- void drawColumns (QPainter *, const QwtScaleMap &xMap, const QwtScaleMap &yMap, int from, int to)
- void drawOutline (QPainter *, const QwtScaleMap &xMap, const QwtScaleMap &yMap, int from, int to) const
- void drawLines (QPainter *, const QwtScaleMap &xMap, const QwtScaleMap &yMap, int from, int to) const

12.74.1 Detailed Description

QwtPlotHistogram represents a series of samples, where an interval is associated with a value ($y = f([x_1, x_2])$).

The representation depends on the style() and an optional symbol() that is displayed for each interval.

Note

The term "histogram" is used in a different way in the areas of digital image processing and statistics. Wikipedia introduces the terms "image histogram" and "color histogram" to avoid confusions. While "image histograms" can be displayed by a QwtPlotCurve there is no applicable plot item for a "color histogram" yet.

See Also

QwtPlotBarChart, QwtPlotMultiBarChart

12.74.2 Member Enumeration Documentation

12.74.2.1 enum QwtPlotHistogram::HistogramStyle

Histogram styles. The default style is QwtPlotHistogram::Columns.

See Also

setStyle(), style(), setSymbol(), symbol(), setBaseline()

Enumerator

Outline Draw an outline around the area, that is build by all intervals using the pen() and fill it with the brush(). The outline style requires, that the intervals are in increasing order and not overlapping.

Columns Draw a column for each interval. When a symbol() has been set the symbol is used otherwise the column is displayed as plain rectangle using pen() and brush().

Lines Draw a simple line using the pen() for each interval.

UserStyle Styles >= UserStyle are reserved for derived classes that overload drawSeries() with additional application specific ways to display a histogram.

12.74.3 Constructor & Destructor Documentation

12.74.3.1 QwtPlotHistogram::QwtPlotHistogram (const QString & title = QString::null) [explicit]

Constructor

Parameters

| title | Title of the histogram. |
|-------|-------------------------|
|-------|-------------------------|

12.74.3.2 QwtPlotHistogram::QwtPlotHistogram (const QwtText & title) [explicit]

Constructor

Parameters

| title | Title of the histogram. |
|-------|-------------------------|

12.74.4 Member Function Documentation

12.74.4.1 double QwtPlotHistogram::baseline () const

Returns

Value of the baseline

See Also

setBaseline()

12.74.4.2 QRectF QwtPlotHistogram::boundingRect() const [virtual]

Returns

Bounding rectangle of all samples. For an empty series the rectangle is invalid.

Reimplemented from QwtPlotSeriesItem.

12.74.4.3 const QBrush & QwtPlotHistogram::brush () const

Returns

Brush used in a style() depending way.

See Also

setPen(), brush()

12.74.4.4 QwtColumnRect QwtPlotHistogram::columnRect (const QwtIntervalSample & sample, const QwtScaleMap & xMap, const QwtScaleMap & yMap) const [protected], [virtual]

Calculate the area that is covered by a sample

Parameters

| sample | Sample |
|--------|---------------------------------------|
| хМар | Maps x-values into pixel coordinates. |
| уМар | Maps y-values into pixel coordinates. |

Returns

Rectangle, that is covered by a sample

12.74.4.5 void QwtPlotHistogram::drawColumn (QPainter * painter, const QwtColumnRect & rect, const QwtIntervalSample & sample) const [protected], [virtual]

Draw a column for a sample in Columns style().

When a symbol() has been set the symbol is used otherwise the column is displayed as plain rectangle using pen() and brush().

Parameters

| painter | Painter |
|---------|---|
| rect | Rectangle where to paint the column in paint device coordinates |
| sample | Sample to be displayed |

Note

In applications, where different intervals need to be displayed in a different way (f.e different colors or even using different symbols) it is recommended to overload drawColumn().

12.74.4.6 void QwtPlotHistogram::drawColumns (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, int from, int to) const [protected]

Draw a histogram in Columns style()

Parameters

| painter | Painter |
|---------|--|
| хМар | Maps x-values into pixel coordinates. |
| уМар | Maps y-values into pixel coordinates. |
| from | Index of the first sample to be painted |
| to | Index of the last sample to be painted. If to < 0 the histogram will be painted to its last point. |

See Also

setStyle(), style(), setSymbol(), drawColumn()

12.74.4.7 void QwtPlotHistogram::drawLines (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, int from, int to) const [protected]

Draw a histogram in Lines style()

Parameters

| painter | Painter |
|---------|--|
| хМар | Maps x-values into pixel coordinates. |
| уМар | Maps y-values into pixel coordinates. |
| from | Index of the first sample to be painted |
| to | Index of the last sample to be painted. If to $<$ 0 the histogram will be painted to its last point. |

See Also

setStyle(), style(), setPen()

12.74.4.8 void QwtPlotHistogram::drawOutline (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, int from, int to) const [protected]

Draw a histogram in Outline style()

Parameters

| painter | Painter |
|---------|---------------------------------------|
| хМар | Maps x-values into pixel coordinates. |
| уМар | Maps y-values into pixel coordinates. |

| from | Index of the first sample to be painted |
|------|--|
| to | Index of the last sample to be painted. If to $<$ 0 the histogram will be painted to its last point. |

See Also

setStyle(), style()

Warning

The outline style requires, that the intervals are in increasing order and not overlapping.

12.74.4.9 void QwtPlotHistogram::drawSeries (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [virtual]

Draw a subset of the histogram samples

Parameters

| painter | Painter |
|------------|--|
| хМар | Maps x-values into pixel coordinates. |
| уМар | Maps y-values into pixel coordinates. |
| canvasRect | Contents rectangle of the canvas |
| from | Index of the first sample to be painted |
| to | Index of the last sample to be painted. If to $<$ 0 the series will be painted to its last sample. |

See Also

drawOutline(), drawLines(), drawColumns

Implements QwtPlotSeriesItem.

12.74.4.10 QwtGraphic QwtPlotHistogram::legendlcon (int index, const QSizeF & size) const [virtual]

A plain rectangle without pen using the brush()

Parameters

| index | Index of the legend entry (ignored as there is only one) |
|-------|--|
| size | Icon size |

Returns

A graphic displaying the icon

See Also

QwtPlotItem::setLegendIconSize(), QwtPlotItem::legendData()

Reimplemented from QwtPlotItem.

12.74.4.11 const QPen & QwtPlotHistogram::pen () const

Returns

Pen used in a style() depending way.

See Also

setPen(), brush()

12.74.4.12 int QwtPlotHistogram::rtti() const [virtual]

Returns

QwtPlotItem::Rtti_PlotHistogram

Reimplemented from QwtPlotItem.

12.74.4.13 void QwtPlotHistogram::setBaseline (double value)

Set the value of the baseline.

Each column representing an QwtIntervalSample is defined by its interval and the interval between baseline and the value of the sample.

The default value of the baseline is 0.0.

Parameters

| value | Value of the baseline |
|-------|--------------------------|
| | Talado di lito daddiii d |

See Also

baseline()

12.74.4.14 void QwtPlotHistogram::setBrush (const QBrush & brush)

Assign a brush, that is used in a style() depending way.

Parameters

| brush | New brush |
|-------|-----------|
|-------|-----------|

See Also

pen(), brush()

12.74.4.15 void QwtPlotHistogram::setPen (const QColor & color, qreal width = 0.0, Qt::PenStyle style = Qt::SolidLine)

Build and assign a pen

In Qt5 the default pen width is 1.0 (0.0 in Qt4) what makes it non cosmetic (see QPen::isCosmetic()). This method has been introduced to hide this incompatibility.

Parameters

| color | Pen color |
|-------|-----------|
| width | Pen width |
| style | Pen style |

See Also

pen(), brush()

12.74.4.16 void QwtPlotHistogram::setPen (const QPen & pen)

Assign a pen, that is used in a style() depending way.

Parameters

| pen | New pen |
|-----|---------|
|-----|---------|

See Also

pen(), brush()

12.74.4.17 void QwtPlotHistogram::setSamples (const QVector < QwtIntervalSample > & samples)

Initialize data with an array of samples.

Parameters

| samples | Vector of points |
|---------|------------------|

12.74.4.18 void QwtPlotHistogram::setSamples (QwtSeriesData < QwtIntervalSample > * data)

Assign a series of samples

setSamples() is just a wrapper for setData() without any additional value - beside that it is easier to find for the developer.

Parameters

| data | Data |
|------|--------|
| uala | i Dala |
| | |

Warning

The item takes ownership of the data object, deleting it when its not used anymore.

12.74.4.19 void QwtPlotHistogram::setStyle (HistogramStyle style)

Set the histogram's drawing style

Parameters

| style | Histogram style |
|-------|-----------------|

See Also

HistogramStyle, style()

12.74.4.20 void QwtPlotHistogram::setSymbol (const QwtColumnSymbol * symbol)

Assign a symbol.

In Column style an optional symbol can be assigned, that is responsible for displaying the rectangle that is defined by the interval and the distance between baseline() and value. When no symbol has been defined the area is displayed as plain rectangle using pen() and brush().

See Also

```
style(), symbol(), drawColumn(), pen(), brush()
```

Note

In applications, where different intervals need to be displayed in a different way (f.e different colors or even using different symbols) it is recommended to overload drawColumn().

12.74.4.21 QwtPlotHistogram::HistogramStyle QwtPlotHistogram::style () const

Returns

Style of the histogram

See Also

HistogramStyle, setStyle()

12.74.4.22 const QwtColumnSymbol * QwtPlotHistogram::symbol () const

Returns

Current symbol or NULL, when no symbol has been assigned

See Also

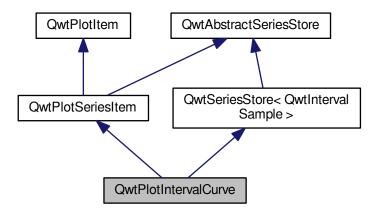
setSymbol()

12.75 QwtPlotIntervalCurve Class Reference

QwtPlotIntervalCurve represents a series of samples, where each value is associated with an interval ([y1,y2] = f(x)).

#include <qwt_plot_intervalcurve.h>

Inheritance diagram for QwtPlotIntervalCurve:



Public Types

- enum CurveStyle { NoCurve, Tube, UserCurve = 100 }
 - Curve styles. The default setting is QwtPlotIntervalCurve::Tube.
- enum PaintAttribute { ClipPolygons = 0x01, ClipSymbol = 0x02 }
- typedef QFlags< PaintAttribute > PaintAttributes

Paint attributes.

Public Member Functions

- QwtPlotIntervalCurve (const QString &title=QString::null)
- QwtPlotIntervalCurve (const QwtText &title)
- virtual ~QwtPlotIntervalCurve ()

Destructor.

- · virtual int rtti () const
- void setPaintAttribute (PaintAttribute, bool on=true)
- · bool testPaintAttribute (PaintAttribute) const
- void setSamples (const QVector < QwtIntervalSample > &)
- void setSamples (QwtSeriesData< QwtIntervalSample > *)
- void setPen (const QColor &, greal width=0.0, Qt::PenStyle=Qt::SolidLine)
- void setPen (const QPen &)

Assign a pen.

- const QPen & pen () const
- void setBrush (const QBrush &)
- · const QBrush & brush () const
- void setStyle (CurveStyle style)
- CurveStyle style () const
- void setSymbol (const QwtIntervalSymbol *)
- const QwtIntervalSymbol * symbol () const
- virtual void drawSeries (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int from, int to) const
- · virtual QRectF boundingRect () const
- virtual QwtGraphic legendlcon (int index, const QSizeF &) const

Protected Member Functions

• void init ()

Initialize internal members.

- virtual void drawTube (QPainter *, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int from, int to) const
- virtual void drawSymbols (QPainter *, const QwtIntervalSymbol &, const QwtScaleMap &xMap, const Qwt-ScaleMap &yMap, const QRectF &canvasRect, int from, int to) const

12.75.1 Detailed Description

QwtPlotIntervalCurve represents a series of samples, where each value is associated with an interval ([y1,y2] = f(x)).

The representation depends on the style() and an optional symbol() that is displayed for each interval. QwtPlot-IntervalCurve might be used to display error bars or the area between 2 curves.

12.75.2 Member Enumeration Documentation

12.75.2.1 enum QwtPlotIntervalCurve::CurveStyle

Curve styles. The default setting is QwtPlotIntervalCurve::Tube.

See Also

setStyle(), style()

Enumerator

NoCurve Don't draw a curve. Note: This doesn't affect the symbols.

Tube Build 2 curves from the upper and lower limits of the intervals and draw them with the pen(). The area between the curves is filled with the brush().

UserCurve Styles >= QwtPlotIntervalCurve::UserCurve are reserved for derived classes that overload draw-Series() with additional application specific curve types.

12.75.2.2 enum QwtPlotIntervalCurve::PaintAttribute

Attributes to modify the drawing algorithm.

See Also

setPaintAttribute(), testPaintAttribute()

Enumerator

ClipPolygons Clip polygons before painting them. In situations, where points are far outside the visible area (f.e when zooming deep) this might be a substantial improvement for the painting performance.

ClipSymbol Check if a symbol is on the plot canvas before painting it.

12.75.3 Constructor & Destructor Documentation

12.75.3.1 QwtPlotIntervalCurve::QwtPlotIntervalCurve (const QString & title = QString::null) [explicit]

Constructor

Parameters

| title | Title of the curve |
|-------|--------------------|

12.75.3.2 QwtPlotIntervalCurve::QwtPlotIntervalCurve (const QwtText & title) [explicit]

Constructor

Parameters

| title | Title of the curve |
|-------|--------------------|
|-------|--------------------|

12.75.4 Member Function Documentation

12.75.4.1 QRectF QwtPlotIntervalCurve::boundingRect() const [virtual]

Returns

Bounding rectangle of all samples. For an empty series the rectangle is invalid.

Reimplemented from QwtPlotSeriesItem.

12.75.4.2 const QBrush & QwtPlotIntervalCurve::brush () const

Returns

Brush used to fill the area in Tube style()

See Also

setBrush(), setStyle(), CurveStyle

12.75.4.3 void QwtPlotIntervalCurve::drawSeries (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [virtual]

Draw a subset of the samples

Parameters

| painter | Painter |
|------------|--|
| хМар | Maps x-values into pixel coordinates. |
| уМар | Maps y-values into pixel coordinates. |
| canvasRect | Contents rectangle of the canvas |
| from | Index of the first sample to be painted |
| to | Index of the last sample to be painted. If to $<$ 0 the series will be painted to its last sample. |

See Also

drawTube(), drawSymbols()

Implements QwtPlotSeriesItem.

12.75.4.4 void QwtPlotIntervalCurve::drawSymbols (QPainter * painter, const QwtIntervalSymbol & symbol, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [protected], [virtual]

Draw symbols for a subset of the samples

Parameters

| painter | Painter |
|------------|---|
| symbol | Interval symbol |
| хМар | x map |
| уМар | y map |
| canvasRect | Contents rectangle of the canvas |
| from | Index of the first sample to be painted |
| to | Index of the last sample to be painted |

See Also

setSymbol(), drawSeries(), drawTube()

12.75.4.5 void QwtPlotIntervalCurve::drawTube (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [protected], [virtual]

Draw a tube

Builds 2 curves from the upper and lower limits of the intervals and draws them with the pen(). The area between the curves is filled with the brush().

Parameters

| painter | Painter |
|---------|---------------------------------------|
| хМар | Maps x-values into pixel coordinates. |

| уМар | Maps y-values into pixel coordinates. |
|------------|--|
| canvasRect | Contents rectangle of the canvas |
| from | Index of the first sample to be painted |
| to | Index of the last sample to be painted. If to $<$ 0 the series will be painted to its last sample. |

See Also

drawSeries(), drawSymbols()

12.75.4.6 QwtGraphic QwtPlotIntervalCurve::legendlcon (int index, const QSizeF & size) const [virtual]

Returns

Icon for the legend

In case of Tube style() the icon is a plain rectangle filled with the brush(). If a symbol is assigned it is scaled to size.

Parameters

| index | Index of the legend entry (ignored as there is only one) |
|-------|--|
| size | Icon size |

See Also

QwtPlotItem::setLegendIconSize(), QwtPlotItem::legendData()

Reimplemented from QwtPlotItem.

12.75.4.7 const QPen & QwtPlotIntervalCurve::pen () const

Returns

Pen used to draw the lines

See Also

setPen(), brush()

12.75.4.8 int QwtPlotIntervalCurve::rtti () const [virtual]

Returns

QwtPlotItem::Rtti_PlotIntervalCurve

Reimplemented from QwtPlotItem.

12.75.4.9 void QwtPlotIntervalCurve::setBrush (const QBrush & brush)

Assign a brush.

The brush is used to fill the area in Tube style().

Parameters

| brush | Brush |
|-------|-------|

See Also

brush(), pen(), setStyle(), CurveStyle

12.75.4.10 void QwtPlotIntervalCurve::setPaintAttribute (PaintAttribute attribute, bool on = true)

Specify an attribute how to draw the curve

Parameters

| attribute | Paint attribute |
|-----------|-----------------|
| on | On/Off |

See Also

testPaintAttribute()

12.75.4.11 void QwtPlotIntervalCurve::setPen (const QColor & color, qreal width = 0.0, Qt::PenStyle style = Qt::SolidLine)

Build and assign a pen

In Qt5 the default pen width is 1.0 (0.0 in Qt4) what makes it non cosmetic (see QPen::isCosmetic()). This method has been introduced to hide this incompatibility.

Parameters

| color | Pen color |
|-------|-----------|
| width | Pen width |
| style | Pen style |

See Also

pen(), brush()

12.75.4.12 void QwtPlotIntervalCurve::setPen (const QPen & pen)

Assign a pen.

Parameters

| pen | New pen |
|-----|---------|

See Also

pen(), brush()

12.75.4.13 void QwtPlotIntervalCurve::setSamples (const QVector < QwtIntervalSample > & samples)

Initialize data with an array of samples.

Parameters

| samples | Vector of samples |
|---------|-------------------|

12.75.4.14 void QwtPlotIntervalCurve::setSamples (QwtSeriesData < QwtIntervalSample > * data)

Assign a series of samples

setSamples() is just a wrapper for setData() without any additional value - beside that it is easier to find for the developer.

Parameters

| data | Data |
|------|------|

Warning

The item takes ownership of the data object, deleting it when its not used anymore.

12.75.4.15 void QwtPlotIntervalCurve::setStyle ($CurveStyle \ style$)

Set the curve's drawing style

Parameters

style Curve style

See Also

CurveStyle, style()

12.75.4.16 void QwtPlotIntervalCurve::setSymbol (const QwtIntervalSymbol * symbol)

Assign a symbol.

Parameters

symbol Symbol

See Also

symbol()

12.75.4.17 QwtPlotIntervalCurve::CurveStyle QwtPlotIntervalCurve::style () const

Returns

Style of the curve

See Also

setStyle()

12.75.4.18 const QwtIntervalSymbol * QwtPlotIntervalCurve::symbol () const

Returns

Current symbol or NULL, when no symbol has been assigned

See Also

setSymbol()

 $12.75.4.19 \quad bool \ QwtPlotIntervalCurve:: testPaintAttribute \ (\ PaintAttribute \ attribute \) \ const$

Returns

True, when attribute is enabled

See Also

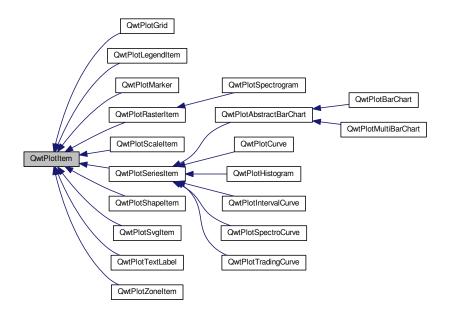
PaintAttribute, setPaintAttribute()

12.76 QwtPlotItem Class Reference

Base class for items on the plot canvas.

#include <qwt_plot_item.h>

Inheritance diagram for QwtPlotItem:



Public Types

```
    enum RttiValues {
    Rtti_PlotItem = 0, Rtti_PlotGrid, Rtti_PlotScale, Rtti_PlotLegend,
    Rtti_PlotMarker, Rtti_PlotCurve, Rtti_PlotSpectroCurve, Rtti_PlotIntervalCurve,
    Rtti_PlotHistogram, Rtti_PlotSpectrogram, Rtti_PlotSVG, Rtti_PlotTradingCurve,
    Rtti_PlotBarChart, Rtti_PlotMultiBarChart, Rtti_PlotShape, Rtti_PlotTextLabel,
    Rtti_PlotZone, Rtti_PlotUserItem = 1000 }
```

Runtime type information.

- enum ItemAttribute { Legend = 0x01, AutoScale = 0x02, Margins = 0x04 }
 - Plot Item Attributes.
- enum ItemInterest { ScaleInterest = 0x01, LegendInterest = 0x02 }

Plot Item Interests.

- enum RenderHint { RenderAntialiased = 0x1 }
 - Render hints.
- typedef QFlags < ItemAttribute > ItemAttributes

Plot Item Attributes.

- $\bullet \ \ type def \ QFlags < ItemInterest > ItemInterests \\$
 - Plot Item Interests.
- typedef QFlags< RenderHint > RenderHints

Render hints.

Public Member Functions

- QwtPlotItem (const QwtText &title=QwtText())
- virtual ~QwtPlotItem ()

Destroy the QwtPlotItem.

void attach (QwtPlot *plot)

Attach the item to a plot.

· void detach ()

This method detaches a QwtPlotItem from any QwtPlot it has been associated with.

QwtPlot * plot () const

Return attached plot.

- void setTitle (const QString &title)
- void setTitle (const QwtText &title)
- const QwtText & title () const
- · virtual int rtti () const
- void setItemAttribute (ItemAttribute, bool on=true)
- · bool testItemAttribute (ItemAttribute) const
- void setItemInterest (ItemInterest, bool on=true)
- · bool testItemInterest (ItemInterest) const
- void setRenderHint (RenderHint, bool on=true)
- bool testRenderHint (RenderHint) const
- void setRenderThreadCount (uint numThreads)
- · uint renderThreadCount () const
- void setLegendIconSize (const QSize &)
- QSize legendlconSize () const
- · double z () const
- void setZ (double z)

Set the z value.

• void show ()

Show the item.

· void hide ()

Hide the item.

- virtual void setVisible (bool)
- bool isVisible () const
- · void setAxes (int xAxis, int yAxis)
- void setXAxis (int axis)
- int xAxis () const

Return xAxis.

- void setYAxis (int axis)
- int yAxis () const

Return yAxis.

- virtual void itemChanged ()
- virtual void legendChanged ()
- virtual void draw (QPainter *painter, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect) const =0

Draw the item.

- · virtual QRectF boundingRect () const
- virtual void getCanvasMarginHint (const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasSize, double &left, double &top, double &right, double &bottom) const

Calculate a hint for the canvas margin.

virtual void updateScaleDiv (const QwtScaleDiv &, const QwtScaleDiv &)

Update the item to changes of the axes scale division.

virtual void updateLegend (const QwtPlotItem *, const QList< QwtLegendData > &)

Update the item to changes of the legend info.

QRectF scaleRect (const QwtScaleMap &, const QwtScaleMap &) const

Calculate the bounding scale rectangle of 2 maps.

• QRectF paintRect (const QwtScaleMap &, const QwtScaleMap &) const

Calculate the bounding paint rectangle of 2 maps.

virtual QList< QwtLegendData > legendData () const

Return all information, that is needed to represent the item on the legend.

virtual QwtGraphic legendlcon (int index, const QSizeF &) const

Protected Member Functions

QwtGraphic defaultIcon (const QBrush &, const QSizeF &) const

Return a default icon from a brush.

12.76.1 Detailed Description

Base class for items on the plot canvas.

A plot item is "something", that can be painted on the plot canvas, or only affects the scales of the plot widget. They can be categorized as:

· Representator

A "Representator" is an item that represents some sort of data on the plot canvas. The different representator classes are organized according to the characteristics of the data:

- QwtPlotMarker Represents a point or a horizontal/vertical coordinate
- QwtPlotCurve Represents a series of points
- QwtPlotSpectrogram (QwtPlotRasterItem) Represents raster data
- ...
- · Decorators

A "Decorator" is an item, that displays additional information, that is not related to any data:

- QwtPlotGrid
- QwtPlotScaleItem
- QwtPlotSvgItem
- ...

Depending on the QwtPlotItem::ItemAttribute flags, an item is included into autoscaling or has an entry on the legend.

Before misusing the existing item classes it might be better to implement a new type of plot item (don't implement a watermark as spectrogram). Deriving a new type of QwtPlotItem primarily means to implement the YourPlotItem::draw() method.

See Also

The cpuplot example shows the implementation of additional plot items.

12.76.2 Member Enumeration Documentation

12.76.2.1 enum QwtPlotItem::ItemAttribute

Plot Item Attributes.

Various aspects of a plot widget depend on the attributes of the attached plot items. If and how a single plot item participates in these updates depends on its attributes.

See Also

setItemAttribute(), testItemAttribute(), ItemInterest

Enumerator

Legend The item is represented on the legend.

AutoScale The boundingRect() of the item is included in the autoscaling calculation as long as its width or height is >= 0.0.

Margins The item needs extra space to display something outside its bounding rectangle.

See Also

getCanvasMarginHint()

12.76.2.2 enum QwtPlotItem::ItemInterest

Plot Item Interests.

Plot items might depend on the situation of the corresponding plot widget. By enabling an interest the plot item will be notified, when the corresponding attribute of the plot widgets has changed.

See Also

setItemAttribute(), testItemAttribute(), ItemInterest

Enumerator

ScaleInterest The item is interested in updates of the scales

See Also

updateScaleDiv()

LegendInterest The item is interested in updates of the legend (of other items) This flag is intended for items, that want to implement a legend for displaying entries of other plot item.

Note

If the plot item wants to be represented on a legend enable QwtPlotItem::Legend instead.

See Also

updateLegend()

12.76.2.3 enum QwtPlotItem::RenderHint

Render hints.

Enumerator

RenderAntialiased Enable antialiasing.

12.76.2.4 enum QwtPlotItem::RttiValues

Runtime type information.

RttiValues is used to cast plot items, without having to enable runtime type information of the compiler.

Enumerator

Rtti_PlotItem Unspecific value, that can be used, when it doesn't matter.

Rtti_PlotGrid For QwtPlotGrid.

Rtti_PlotScale For QwtPlotScaleItem.

Rtti_PlotLegend For QwtPlotLegendItem.

Rtti_PlotMarker For QwtPlotMarker.

Rtti_PlotCurve For QwtPlotCurve.

Rtti_PlotSpectroCurve For QwtPlotSpectroCurve.

Rtti_PlotIntervalCurve For QwtPlotIntervalCurve.

Rtti_PlotHistogram For QwtPlotHistogram.

Rtti_PlotSpectrogram For QwtPlotSpectrogram.

Rtti_PlotSVG For QwtPlotSvgItem.

Rtti_PlotTradingCurve For QwtPlotTradingCurve.

Rtti_PlotBarChart For QwtPlotBarChart.

Rtti_PlotMultiBarChart For QwtPlotMultiBarChart.

Rtti_PlotShape For QwtPlotShapeItem.

Rtti_PlotTextLabel For QwtPlotTextLabel.

Rtti_PlotZone For QwtPlotZoneItem.

Rtti_PlotUserItem Values >= Rtti_PlotUserItem are reserved for plot items not implemented in the Qwt library.

12.76.3 Constructor & Destructor Documentation

12.76.3.1 QwtPlotItem::QwtPlotItem(const QwtText & title = QwtText()) [explicit]

Constructor

Parameters

title | Title of the item

12.76.4 Member Function Documentation

12.76.4.1 void QwtPlotItem::attach (QwtPlot * plot)

Attach the item to a plot.

This method will attach a QwtPlotItem to the QwtPlot argument. It will first detach the QwtPlotItem from any plot from a previous call to attach (if necessary). If a NULL argument is passed, it will detach from any QwtPlot it was attached to.

Parameters

| plot | Plot widget |
|------|-------------|

See Also

detach()

12.76.4.2 QRectF QwtPlotItem::boundingRect() const [virtual]

Returns

An invalid bounding rect: QRectF(1.0, 1.0, -2.0, -2.0)

Note

A width or height < 0.0 is ignored by the autoscaler

Reimplemented in QwtPlotTradingCurve, QwtPlotMarker, QwtPlotIntervalCurve, QwtPlotHistogram, QwtPlotRaster-Item, QwtPlotShapeItem, QwtPlotBarChart, QwtPlotMultiBarChart, QwtPlotZoneItem, QwtPlotSeriesItem, and Qwt-PlotSvgItem.

12.76.4.3 QwtGraphic QwtPlotItem::defaultIcon (const QBrush & brush, const QSizeF & size) const [protected]

Return a default icon from a brush.

The default icon is a filled rectangle used in several derived classes as legendlcon().

Parameters

| brush | Fill brush |
|-------|------------|
| size | Icon size |

Returns

A filled rectangle

12.76.4.4 void QwtPlotItem::detach ()

This method detaches a QwtPlotItem from any QwtPlot it has been associated with.

detach() is equivalent to calling attach(NULL)

See Also

attach()

12.76.4.5 virtual void QwtPlotItem::draw (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect) const [pure virtual]

Draw the item.

Parameters

| painter | Painter |
|------------|--|
| хМар | Maps x-values into pixel coordinates. |
| уМар | Maps y-values into pixel coordinates. |
| canvasRect | Contents rect of the canvas in painter coordinates |

Implemented in QwtPlotMarker, QwtPlotLegendItem, QwtPlotRasterItem, QwtPlotShapeItem, QwtPlotSpectrogram, QwtPlotScaleItem, QwtPlotGrid, QwtPlotTextLabel, QwtPlotZoneItem, QwtPlotSvgItem, and Qwt-PlotSeriesItem.

12.76.4.6 void QwtPlotItem::getCanvasMarginHint (const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, double & left, double & top, double & right, double & bottom) const [virtual]

Calculate a hint for the canvas margin.

When the QwtPlotItem::Margins flag is enabled the plot item indicates, that it needs some margins at the borders of the canvas. This is f.e. used by bar charts to reserve space for displaying the bars.

The margins are in target device coordinates (pixels on screen)

Parameters

| хМар | Maps x-values into pixel coordinates. |
|------------|---|
| уМар | Maps y-values into pixel coordinates. |
| canvasRect | Contents rectangle of the canvas in painter coordinates |
| left | Returns the left margin |
| top | Returns the top margin |
| right | Returns the right margin |
| bottom | Returns the bottom margin |

Returns

The default implementation returns 0 for all margins

See Also

QwtPlot::getCanvasMarginsHint(), QwtPlot::updateCanvasMargins()

Reimplemented in QwtPlotAbstractBarChart.

```
12.76.4.7 bool QwtPlotItem::isVisible ( ) const
Returns
      true if visible
See Also
      setVisible(), show(), hide()
12.76.4.8 void QwtPlotItem::itemChanged() [virtual]
Update the legend and call QwtPlot::autoRefresh() for the parent plot.
See Also
      QwtPlot::legendChanged(), QwtPlot::autoRefresh()
12.76.4.9 void QwtPlotItem::legendChanged() [virtual]
Update the legend of the parent plot.
See Also
      QwtPlot::updateLegend(), itemChanged()
12.76.4.10 QList < QwtLegendData > QwtPlotItem::legendData ( ) const [virtual]
Return all information, that is needed to represent the item on the legend.
Most items are represented by one entry on the legend showing an icon and a text, but f.e. QwtPlotMultiBarChart
displays one entry for each bar.
QwtLegendData is basically a list of QVariants that makes it possible to overload and reimplement legendData() to
return almost any type of information, that is understood by the receiver that acts as the legend.
The default implementation returns one entry with the title() of the item and the legendlcon().
Returns
      Data, that is needed to represent the item on the legend
See Also
      title(), legendlcon(), QwtLegend, QwtPlotLegendltem
Reimplemented in QwtPlotBarChart, and QwtPlotMultiBarChart.
12.76.4.11 QwtGraphic QwtPlotItem::legendlcon( int index, const QSizeF & size ) const [virtual]
Returns
      Icon representing the item on the legend
```

The default implementation returns an invalid icon

Parameters

| index | Index of the legend entry (usually there is only one) |
|-------|---|
| size | Icon size |

See Also

setLegendIconSize(), legendData()

Reimplemented in QwtPlotCurve, QwtPlotTradingCurve, QwtPlotMarker, QwtPlotIntervalCurve, QwtPlotHistogram, QwtPlotBarChart, QwtPlotShapeltem, and QwtPlotMultiBarChart.

12.76.4.12 QSize QwtPlotItem::legendlconSize () const

Returns

Legend icon size

See Also

setLegendIconSize(), legendIcon()

12.76.4.13 QRectF QwtPlotItem::paintRect (const QwtScaleMap & xMap, const QwtScaleMap & yMap) const

Calculate the bounding paint rectangle of 2 maps.

Parameters

| хМар | Maps x-values into pixel coordinates. |
|------|---------------------------------------|
| уМар | Maps y-values into pixel coordinates. |

Returns

Bounding paint rectangle of the scale maps, not normalized

12.76.4.14 uint QwtPlotItem::renderThreadCount () const

Returns

Number of threads to be used for rendering. If numThreads() is set to 0, the system specific ideal thread count is used.

12.76.4.15 int QwtPlotItem::rtti() const [virtual]

Return rtti for the specific class represented. QwtPlotItem is simply a virtual interface class, and base classes will implement this method with specific rtti values so a user can differentiate them.

The rtti value is useful for environments, where the runtime type information is disabled and it is not possible to do a dynamic_cast<...>.

Returns

rtti value

See Also

RttiValues

Reimplemented in QwtPlotCurve, QwtPlotTradingCurve, QwtPlotShapeItem, QwtPlotSpectrogram, QwtPlotInterval-Curve, QwtPlotHistogram, QwtPlotMarker, QwtPlotBarChart, QwtPlotMultiBarChart, QwtPlotLegendItem, QwtPlotScaleItem, QwtPlotTextLabel, QwtPlotSpectroCurve, QwtPlotSvgItem, QwtPlotGrid, and QwtPlotZoneItem.

12.76.4.16 QRectF QwtPlotItem::scaleRect (const QwtScaleMap & xMap, const QwtScaleMap & yMap) const Calculate the bounding scale rectangle of 2 maps.

Parameters

| хМар | Maps x-values into pixel coordinates. |
|------|---------------------------------------|
| уМар | Maps y-values into pixel coordinates. |

Returns

Bounding scale rect of the scale maps, not normalized

12.76.4.17 void QwtPlotItem::setAxes (int xAxis, int yAxis)

Set X and Y axis

The item will painted according to the coordinates of its Axes.

Parameters

| xAxis | X Axis (QwtPlot::xBottom or QwtPlot::xTop) |
|-------|--|
| yAxis | Y Axis (QwtPlot::yLeft or QwtPlot::yRight) |

See Also

setXAxis(), setYAxis(), xAxis(), yAxis(), QwtPlot::Axis

12.76.4.18 void QwtPlotItem::setItemAttribute (ItemAttribute attribute, bool on = true)

Toggle an item attribute

Parameters

| attribute | Attribute type |
|-----------|----------------|
| on | true/false |

See Also

testItemAttribute(), ItemInterest

12.76.4.19 void QwtPlotItem::setItemInterest (ItemInterest interest, bool on = true)

Toggle an item interest

Parameters

| interest | Interest type |
|----------|---------------|
| on | true/false |

See Also

testItemInterest(), ItemAttribute

12.76.4.20 void QwtPlotItem::setLegendIconSize (const QSize & size)

Set the size of the legend icon

The default setting is 8x8 pixels

Parameters

| size | Size |
|------|------|
|------|------|

See Also

legendlconSize(), legendlcon()

12.76.4.21 void QwtPlotItem::setRenderHint (RenderHint hint, bool on = true)

Toggle an render hint

Parameters

| hint | Render hint |
|------|-------------|
| on | true/false |

See Also

testRenderHint(), RenderHint

12.76.4.22 void QwtPlotItem::setRenderThreadCount (uint numThreads)

On multi core systems rendering of certain plot item (f.e QwtPlotRasterItem) can be done in parallel in several threads.

The default setting is set to 1.

Parameters

| numThreads | Number of threads to be used for rendering. If numThreads is set to 0, the system specific |
|------------|--|
| | ideal thread count is used. |

The default thread count is 1 (= no additional threads)

12.76.4.23 void QwtPlotItem::setTitle (const QString & title)

Set a new title

Parameters

| title | Title |
|-------|-------|
|-------|-------|

See Also

title()

12.76.4.24 void QwtPlotItem::setTitle (const QwtText & title)

Set a new title

Parameters

| title Title | |
|-------------|--|
|-------------|--|

See Also

title()

12.76.4.25 void QwtPlotItem::setVisible (bool on) [virtual]

Show/Hide the item

Parameters

on Show if true, otherwise hide

See Also

isVisible(), show(), hide()

12.76.4.26 void QwtPlotItem::setXAxis (int axis)

Set the X axis

The item will painted according to the coordinates its Axes.

Parameters

axis X Axis (QwtPlot::xBottom or QwtPlot::xTop)

See Also

setAxes(), setYAxis(), xAxis(), QwtPlot::Axis

12.76.4.27 void QwtPlotItem::setYAxis (int axis)

Set the Y axis

The item will painted according to the coordinates its Axes.

Parameters

axis Y Axis (QwtPlot::yLeft or QwtPlot::yRight)

See Also

setAxes(), setXAxis(), yAxis(), QwtPlot::Axis

12.76.4.28 void QwtPlotItem::setZ (double z)

Set the z value.

Plot items are painted in increasing z-order.

Parameters

z Z-value

See Also

z(), QwtPlotDict::itemList()

12.76.4.29 bool QwtPlotItem::testItemAttribute (ItemAttribute attribute) const

Test an item attribute

Parameters

attribute | Attribute type

Returns

true/false

See Also

setItemAttribute(), ItemInterest

12.76.4.30 bool QwtPlotItem::testItemInterest (ItemInterest interest) const

Test an item interest

Parameters

| interest | Interest type |
|----------|---------------|
|----------|---------------|

Returns

true/false

See Also

setItemInterest(), ItemAttribute

12.76.4.31 bool QwtPlotItem::testRenderHint (RenderHint hint) const

Test a render hint

Parameters

| hint | Render hint |
|------|-------------|
|------|-------------|

Returns

true/false

See Also

setRenderHint(), RenderHint

12.76.4.32 const QwtText & QwtPlotItem::title () const

Returns

Title of the item

See Also

setTitle()

12.76.4.33 void QwtPlotItem::updateLegend (const QwtPlotItem * item, const QList< QwtLegendData > & data) [virtual]

Update the item to changes of the legend info.

Plot items that want to display a legend (not those, that want to be displayed on a legend !) will have to implement updateLegend().

updateLegend() is only called when the LegendInterest interest is enabled. The default implementation does nothing.

Parameters

| item | Plot item to be displayed on a legend |
|------|--|
| data | Attributes how to display item on the legend |

See Also

QwtPlotLegendItem

Note

Plot items, that want to be displayed on a legend need to enable the QwtPlotItem::Legend flag and to implement legendData() and legendIcon()

Reimplemented in QwtPlotLegendItem.

12.76.4.34 void QwtPlotItem::updateScaleDiv (const QwtScaleDiv & xScaleDiv, const QwtScaleDiv & yScaleDiv)
[virtual]

Update the item to changes of the axes scale division.

Update the item, when the axes of plot have changed. The default implementation does nothing, but items that depend on the scale division (like QwtPlotGrid()) have to reimplement updateScaleDiv()

updateScaleDiv() is only called when the ScaleInterest interest is enabled. The default implementation does nothing.

Parameters

| xScaleDiv | Scale division of the x-axis |
|-----------|------------------------------|
| yScaleDiv | Scale division of the y-axis |

See Also

QwtPlot::updateAxes(), ScaleInterest

Reimplemented in QwtPlotScaleItem, QwtPlotGrid, and QwtPlotSeriesItem.

12.76.4.35 double QwtPlotItem::z () const

Plot items are painted in increasing z-order.

Returns

setZ(), QwtPlotDict::itemList()

12.77 QwtPlotLayout Class Reference

Layout engine for QwtPlot.

```
#include <qwt_plot_layout.h>
```

Public Types

- enum Option {
 AlignScales = 0x01, IgnoreScrollbars = 0x02, IgnoreFrames = 0x04, IgnoreLegend = 0x08,
 IgnoreTitle = 0x10, IgnoreFooter = 0x20 }
- typedef QFlags < Option > Options

Layout options.

Public Member Functions

• QwtPlotLayout ()

Constructor.

virtual ~QwtPlotLayout ()

Destructor.

- void setCanvasMargin (int margin, int axis=-1)
- int canvasMargin (int axis) const
- void setAlignCanvasToScales (bool)

Set the align-canvas-to-axis-scales flag for all axes.

- void setAlignCanvasToScale (int axisId, bool)
- bool alignCanvasToScale (int axisId) const
- void setSpacing (int)
- int spacing () const

void setLegendPosition (QwtPlot::LegendPosition pos, double ratio)

Specify the position of the legend.

void setLegendPosition (QwtPlot::LegendPosition pos)

Specify the position of the legend.

- · QwtPlot::LegendPosition legendPosition () const
- void setLegendRatio (double ratio)
- double legendRatio () const
- virtual QSize minimumSizeHint (const QwtPlot *) const
- virtual void activate (const QwtPlot *, const QRectF &rect, Options options=0x00)

Recalculate the geometry of all components.

- virtual void invalidate ()
- QRectF titleRect () const
- QRectF footerRect () const
- QRectF legendRect () const
- QRectF scaleRect (int axis) const
- · QRectF canvasRect () const

Protected Member Functions

void setTitleRect (const QRectF &)

Set the geometry for the title.

void setFooterRect (const QRectF &)

Set the geometry for the footer.

void setLegendRect (const QRectF &)

Set the geometry for the legend.

void setScaleRect (int axis, const QRectF &)

Set the geometry for an axis.

void setCanvasRect (const QRectF &)

Set the geometry for the canvas.

- QRectF layoutLegend (Options options, const QRectF &) const
- QRectF alignLegend (const QRectF &canvasRect, const QRectF &legendRect) const
- void expandLineBreaks (Options options, const QRectF &rect, int &dimTitle, int &dimFooter, int dimAxes[Qwt-Plot::axisCnt]) const
- void alignScales (Options options, QRectF &canvasRect, QRectF scaleRect[QwtPlot::axisCnt]) const

12.77.1 Detailed Description

Layout engine for QwtPlot.

It is used by the QwtPlot widget to organize its internal widgets or by QwtPlot::print() to render its content to a QPaintDevice like a QPrinter, QPixmap/QImage or QSvgRenderer.

See Also

QwtPlot::setPlotLayout()

12.77.2 Member Enumeration Documentation

12.77.2.1 enum QwtPlotLayout::Option

Options to configure the plot layout engine

See Also

activate(), QwtPlotRenderer

Enumerator

AlignScales Unused.

IgnoreScrollbars Ignore the dimension of the scrollbars. There are no scrollbars, when the plot is not rendered to widgets.

IgnoreFrames Ignore all frames.

IgnoreLegend Ignore the legend.

IgnoreTitle Ignore the title.

IgnoreFooter Ignore the footer.

12.77.3 Member Function Documentation

12.77.3.1 void QwtPlotLayout::activate (const QwtPlot * plot, const QRectF & plotRect, Options options = 0×00) [virtual]

Recalculate the geometry of all components.

Parameters

| plot | Plot to be layout |
|----------|---|
| plotRect | Rectangle where to place the components |
| options | Layout options |

See Also

invalidate(), titleRect(), footerRect() legendRect(), scaleRect(), canvasRect()

12.77.3.2 bool QwtPlotLayout::alignCanvasToScale (int axisId) const

Return the align-canvas-to-axis-scales setting. The canvas may:

- · extend beyond the axis scale ends to maximize its size
- align with the axis scale ends to control its size.

Parameters

| axisId | Axis index |
|--------|------------|

Returns

align-canvas-to-axis-scales setting

See Also

setAlignCanvasToScale(), setAlignCanvasToScale(), setCanvasMargin()

12.77.3.3 QRectF QwtPlotLayout::alignLegend (const QRectF & canvasRect, const QRectF & legendRect) const [protected]

Align the legend to the canvas

Parameters

| canvasRect | Geometry of the canvas |
|------------|---------------------------------|
| legendRect | Maximum geometry for the legend |

Returns

Geometry for the aligned legend

12.77.3.4 void QwtPlotLayout::alignScales (Options options, QRectF & canvasRect, QRectF scaleRect[QwtPlot::axisCnt]) const [protected]

Align the ticks of the axis to the canvas borders using the empty corners.

Parameters

| options | Layout options |
|------------|-------------------------------------|
| canvasRect | Geometry of the canvas (IN/OUT) |
| scaleRect | Geometries of the scales (IN/OUT) |

See Also

Options

12.77.3.5 int QwtPlotLayout::canvasMargin (int axisId) const

Parameters

| axisld | Axis index |
|--------|------------|

Returns

Margin around the scale tick borders

See Also

setCanvasMargin()

12.77.3.6 QRectF QwtPlotLayout::canvasRect () const

Returns

Geometry for the canvas

See Also

activate(), invalidate()

12.77.3.7 void QwtPlotLayout::expandLineBreaks (Options options, const QRectF & rect, int & dimTitle, int & dimFooter, int dimAxis[QwtPlot::axisCnt]) const [protected]

Expand all line breaks in text labels, and calculate the height of their widgets in orientation of the text.

Parameters

| options | Options how to layout the legend |
|-----------|--|
| rect | Bounding rectangle for title, footer, axes and canvas. |
| dimTitle | Expanded height of the title widget |
| dimFooter | Expanded height of the footer widget |
| dimAxis | Expanded heights of the axis in axis orientation. |

See Also

Options

12.77.3.8 QRectF QwtPlotLayout::footerRect () const

Returns

Geometry for the footer

See Also

activate(), invalidate()

12.77.3.9 void QwtPlotLayout::invalidate() [virtual]

Invalidate the geometry of all components.

See Also

activate()

12.77.3.10 QRectF QwtPlotLayout::layoutLegend (Options options, const QRectF & rect) const [protected]

Find the geometry for the legend

Parameters

| options | Options how to layout the legend |
|---------|-------------------------------------|
| rect | Rectangle where to place the legend |

Returns

Geometry for the legend

See Also

Options

12.77.3.11 QwtPlot::LegendPosition QwtPlotLayout::legendPosition () const

Returns

Position of the legend

See Also

setLegendPosition(), QwtPlot::setLegendPosition(), QwtPlot::legendPosition()

12.77.3.12 double QwtPlotLayout::legendRatio () const

Returns

The relative size of the legend in the plot.

See Also

setLegendPosition()

12.77.3.13 QRectF QwtPlotLayout::legendRect () const

Returns

Geometry for the legend

See Also

activate(), invalidate()

12.77.3.14 QSize QwtPlotLayout::minimumSizeHint (const QwtPlot * plot) const [virtual]

Returns

Minimum size hint

Parameters

| plot | Plot widget |
|------|-------------|
| , | |

See Also

QwtPlot::minimumSizeHint()

12.77.3.15 QRectF QwtPlotLayout::scaleRect (int axis) const

Parameters

```
axis Axis index
```

Returns

Geometry for the scale

See Also

activate(), invalidate()

12.77.3.16 void QwtPlotLayout::setAlignCanvasToScale (int axisId, bool on)

Change the align-canvas-to-axis-scales setting. The canvas may:

- · extend beyond the axis scale ends to maximize its size,
- · align with the axis scale ends to control its size.

The axisId parameter is somehow confusing as it identifies a border of the plot and not the axes, that are aligned. F.e when QwtPlot::yLeft is set, the left end of the the x-axes (QwtPlot::xTop, QwtPlot::xBottom) is aligned.

Parameters

| axisld | Axis index |
|--------|---|
| on | New align-canvas-to-axis-scales setting |

See Also

setCanvasMargin(), alignCanvasToScale(), setAlignCanvasToScales()

Warning

In case of on == true canvasMargin() will have no effect

12.77.3.17 void QwtPlotLayout::setAlignCanvasToScales (bool on)

Set the align-canvas-to-axis-scales flag for all axes.

Parameters

| on | True/False |
|----|------------|
|----|------------|

See Also

setAlignCanvasToScale(), alignCanvasToScale()

12.77.3.18 void QwtPlotLayout::setCanvasMargin (int margin, int axis = -1)

Change a margin of the canvas. The margin is the space above/below the scale ticks. A negative margin will be set to -1, excluding the borders of the scales.

Parameters

| margin | New margin |
|--------|---|
| axis | One of QwtPlot::Axis. Specifies where the position of the margin1 means margin at all |
| | borders. |

See Also

canvasMargin()

Warning

The margin will have no effect when alignCanvasToScale() is true

12.77.3.19 void QwtPlotLayout::setCanvasRect (const QRectF & rect) [protected]

Set the geometry for the canvas.

This method is intended to be used from derived layouts overloading activate()

See Also

canvasRect(), activate()

12.77.3.20 void QwtPlotLayout::setFooterRect (const QRectF & rect) [protected]

Set the geometry for the footer.

This method is intended to be used from derived layouts overloading activate()

See Also

footerRect(), activate()

12.77.3.21 void QwtPlotLayout::setLegendPosition (QwtPlot::LegendPosition pos, double ratio)

Specify the position of the legend.

Parameters

| pos | The legend's position. |
|-------|---|
| ratio | Ratio between legend and the bounding rectangle of title, footer, canvas and axes. The |
| | legend will be shrunk if it would need more space than the given ratio. The ratio is limited to $]0.0.1.0]$. In case of $<=0.0$ it will be reset to the default ratio. The default vertical/horizontal ratio is $0.33/0.5$. |

See Also

QwtPlot::setLegendPosition()

12.77.3.22 void QwtPlotLayout::setLegendPosition (QwtPlot::LegendPosition pos)

Specify the position of the legend.

Parameters

| ı | pos | The legend's position. Valid values are QwtPlot::LeftLegend, QwtPlot::Right- |
|---|-----|--|
| | | Legend, QwtPlot::TopLegend, QwtPlot::BottomLegend. |

See Also

QwtPlot::setLegendPosition()

12.77.3.23 void QwtPlotLayout::setLegendRatio (double ratio)

Specify the relative size of the legend in the plot

Parameters

| ratio | Ratio between legend and the bounding rectangle of title, footer, canvas and axes. The |
|-------|--|
| | legend will be shrunk if it would need more space than the given ratio. The ratio is limited to |
| | $]0.0.1.0]$. In case of ≤ 0.0 it will be reset to the default ratio. The default vertical/horizontal |
| | ratio is 0.33/0.5. |

12.77.3.24 void QwtPlotLayout::setLegendRect (const QRectF & rect) [protected]

Set the geometry for the legend.

This method is intended to be used from derived layouts overloading activate()

Parameters

| rect | Rectangle for the legend |
|------|--------------------------|

See Also

legendRect(), activate()

12.77.3.25 void QwtPlotLayout::setScaleRect (int axis, const QRectF & rect) [protected]

Set the geometry for an axis.

This method is intended to be used from derived layouts overloading activate()

Parameters

| axis | Axis index |
|------|-------------------------|
| rect | Rectangle for the scale |

See Also

```
scaleRect(), activate()
```

12.77.3.26 void QwtPlotLayout::setSpacing (int spacing)

Change the spacing of the plot. The spacing is the distance between the plot components.

Parameters

| | A1 · |
|---------|----------------|
| spacing | New spacing |
| Spacing | |
| opaonig | 11011 opaoling |

See Also

```
setCanvasMargin(), spacing()
```

12.77.3.27 void QwtPlotLayout::setTitleRect (const QRectF & rect) [protected]

Set the geometry for the title.

This method is intended to be used from derived layouts overloading activate()

See Also

```
titleRect(), activate()
```

12.77.3.28 int QwtPlotLayout::spacing () const

Returns

Spacing

See Also

margin(), setSpacing()

12.77.3.29 QRectF QwtPlotLayout::titleRect () const

Returns

Geometry for the title

See Also

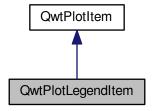
activate(), invalidate()

12.78 QwtPlotLegendItem Class Reference

A class which draws a legend inside the plot canvas.

```
#include <qwt_plot_legenditem.h>
```

Inheritance diagram for QwtPlotLegendItem:



Public Types

enum BackgroundMode { LegendBackground, ItemBackground }
 Background mode.

Public Member Functions

QwtPlotLegendItem ()

Constructor.

virtual ~QwtPlotLegendItem ()

Destructor.

- · virtual int rtti () const
- void setAlignment (Qt::Alignment)

Set the alignmnet.

- Qt::Alignment alignment () const
- void setMaxColumns (uint)

Limit the number of columns.

- uint maxColumns () const
- void setMargin (int)

Set the margin around legend items.

- int margin () const
- void setSpacing (int)

Set the spacing between the legend items.

- int spacing () const
- void setItemMargin (int)
- int itemMargin () const
- void setItemSpacing (int)
- int itemSpacing () const
- void setFont (const QFont &)
- · QFont font () const
- void setBorderDistance (int numPixels)

Set the margin between the legend and the canvas border.

- int borderDistance () const
- void setBorderRadius (double)
- · double borderRadius () const
- void setBorderPen (const QPen &)

- QPen borderPen () const
- void setBackgroundBrush (const QBrush &)

Set the background brush.

- QBrush backgroundBrush () const
- void setBackgroundMode (BackgroundMode)

Set the background mode.

- BackgroundMode backgroundMode () const
- void setTextPen (const QPen &)

Set the pen for drawing text labels.

- QPen textPen () const
- virtual void draw (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &rect) const
- void clearLegend ()

Remove all items from the legend.

- virtual void updateLegend (const QwtPlotItem *, const QList< QwtLegendData > &)
- · virtual QRect geometry (const QRectF &canvasRect) const
- virtual QSize minimumSize (const QwtLegendData &) const
- virtual int heightForWidth (const QwtLegendData &, int w) const
- QList< const QwtPlotItem * > plotItems () const
- QList< QRect > legendGeometries (const QwtPlotItem *) const

Protected Member Functions

- virtual void drawLegendData (QPainter *painter, const QwtPlotItem *, const QwtLegendData &, const QRect-F &) const
- virtual void drawBackground (QPainter *, const QRectF &rect) const

12.78.1 Detailed Description

A class which draws a legend inside the plot canvas.

QwtPlotLegendItem can be used to draw a inside the plot canvas. It can be used together with a QwtLegend or instead of it to have more space for the plot canvas.

In opposite to QwtLegend the legend item is not interactive. To identify mouse clicks on a legend item an event filter needs to be installed catching mouse events ob the plot canvas. The geometries of the legend items are available using legendGeometries().

The legend item is aligned to plot canvas according to its alignment() flags. It might have a background for the complete legend (usually semi transparent) or for each legend item.

Note

An external QwtLegend with a transparent background on top the plot canvas might be another option with a similar effect.

12.78.2 Member Enumeration Documentation

12.78.2.1 enum QwtPlotLegendItem::BackgroundMode

Background mode.

Depending on the mode the complete legend or each item might have an background.

The default setting is LegendBackground.

```
See Also
      setBackgroundMode(), setBackgroundBrush(), drawBackground()
Enumerator
     LegendBackground The legend has a background.
     ItemBackground Each item has a background.
 12.78.3 Member Function Documentation
12.78.3.1 Qt::Alignment QwtPlotLegendItem::alignment ( ) const
 Returns
      Alignment flags
 See Also
      setAlignment()
12.78.3.2 QBrush QwtPlotLegendItem::backgroundBrush ( ) const
 Returns
      Brush is used to fill the background
 See Also
      setBackgroundBrush(), backgroundMode(), drawBackground()
12.78.3.3 QwtPlotLegendItem::BackgroundMode QwtPlotLegendItem::backgroundMode ( ) const
 Returns
      backgroundMode
 See Also
      setBackgroundMode(), backgroundBrush(), drawBackground()
 12.78.3.4 int QwtPlotLegendItem::borderDistance ( ) const
 Returns
      Margin between the legend and the canvas border
 See Also
      margin()
12.78.3.5 QPen QwtPlotLegendItem::borderPen ( ) const
 Returns
      Pen for drawing the border
 See Also
      setBorderPen(), backgroundBrush()
```

12.78.3.6 double QwtPlotLegendItem::borderRadius () const

Returns

Radius of the border

See Also

setBorderRadius(), setBorderPen()

12.78.3.7 void QwtPlotLegendItem::draw (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect) const [virtual]

Draw the legend

Parameters

| | painter | Painter |
|---|------------|---|
| | хМар | x Scale Map |
| ſ | уМар | y Scale Map |
| Ī | canvasRect | Contents rectangle of the canvas in painter coordinates |

Implements QwtPlotItem.

12.78.3.8 void QwtPlotLegendItem::drawBackground (QPainter * painter, const QRectF & rect) const [protected], [virtual]

Draw a rounded rect

Parameters

| pair | ter | Painter |
|------|-----|--------------------|
| , | ect | Bounding rectangle |

See Also

setBorderRadius(), setBorderPen(), setBackgroundBrush(), setBackgroundMode()

12.78.3.9 void QwtPlotLegendItem::drawLegendData (QPainter * painter, const QwtPlotItem * plotItem, const QwtLegendData & data, const QRectF & rect) const [protected], [virtual]

Draw an entry on the legend

Parameters

| | painter | Qt Painter |
|---|----------|-------------------------------------|
| Ī | plotItem | Plot item, represented by the entry |
| | data | Attributes of the legend entry |
| Ī | rect | Bounding rectangle for the entry |

12.78.3.10 QFont QwtPlotLegendItem::font () const

Returns

Font used for drawing the text label

See Also

setFont()

12.78.3.11 QRect QwtPlotLegendItem::geometry (const QRectF & canvasRect) const [virtual]

Calculate the geometry of the legend on the canvas

Parameters

| canvasRect | Geometry of the canvas |
|------------|------------------------|

Returns

Geometry of the legend

12.78.3.12 int QwtPlotLegendItem::heightForWidth (const QwtLegendData & data, int width) const [virtual]

Returns

The preferred height, for a width.

Parameters

| data | Attributes of the legend entry |
|-------|--------------------------------|
| width | Width |

12.78.3.13 int QwtPlotLegendItem::itemMargin () const

Returns

Margin around each item

See Also

setItemMargin(), itemSpacing(), margin(), spacing()

12.78.3.14 int QwtPlotLegendItem::itemSpacing () const

Returns

Spacing inside of each item

See Also

setItemSpacing(), itemMargin(), margin(), spacing()

12.78.3.15 QList < QRect > QwtPlotLegendItem::legendGeometries (const QwtPlotItem * plotItem) const

Returns

Geometries of the items of a plot item

Note

Usually a plot item has only one entry on the legend

12.78.3.16 int QwtPlotLegendItem::margin () const

Returns

Margin around the legend items

See Also

setMargin(), spacing(), itemMargin(), itemSpacing()

12.78.3.17 uint QwtPlotLegendItem::maxColumns () const

Returns

Maximum number of columns

See Also

maxColumns(), QwtDynGridLayout::maxColumns()

12.78.3.18 QSize QwtPlotLegendItem::minimumSize (const QwtLegendData & data) const [virtual]

Minimum size hint needed to display an entry

Parameters

data Attributes of the legend entry

Returns

Minimum size

12.78.3.19 QList < const QwtPlotItem * > QwtPlotLegendItem::plotItems () const

Returns

All plot items with an entry on the legend

Note

A plot item might have more than one entry on the legend

12.78.3.20 int QwtPlotLegendItem::rtti() const [virtual]

Returns

QwtPlotItem::Rtti_PlotLegend

Reimplemented from QwtPlotItem.

12.78.3.21 void QwtPlotLegendItem::setAlignment (Qt::Alignment alignment)

Set the alignmnet.

Alignment means the position of the legend relative to the geometry of the plot canvas.

Parameters

alignment | Alignment flags

See Also

alignment(), setMaxColumns()

Note

To align a legend with many items horizontally the number of columns need to be limited

12.78.3.22 void QwtPlotLegendItem::setBackgroundBrush (const QBrush & brush)

Set the background brush.

The brush is used to fill the background

Parameters

brush Brush

See Also

backgroundBrush(), setBackgroundMode(), drawBackground()

12.78.3.23 void QwtPlotLegendItem::setBackgroundMode (BackgroundMode mode)

Set the background mode.

Depending on the mode the complete legend or each item might have an background.

The default setting is LegendBackground.

See Also

backgroundMode(), setBackgroundBrush(), drawBackground()

12.78.3.24 void QwtPlotLegendItem::setBorderDistance (int distance)

Set the margin between the legend and the canvas border.

The default setting for the margin is 10 pixels.

Parameters

distance Margin in pixels

See Also

setMargin()

12.78.3.25 void QwtPlotLegendItem::setBorderPen (const QPen & pen)

Set the pen for drawing the border

Parameters

pen Border pen

See Also

borderPen(), setBackgroundBrush()

12.78.3.26 void QwtPlotLegendItem::setBorderRadius (double radius)

Set the radius for the border

Parameters

radius A value <= 0 defines a rectangular border

See Also

borderRadius(), setBorderPen()

12.78.3.27 void QwtPlotLegendItem::setFont (const QFont & font)

Change the font used for drawing the text label

Parameters

font Legend font

See Also

font()

12.78.3.28 void QwtPlotLegendItem::setItemMargin (int margin)

Set the margin around each item

Parameters

margin Margin

See Also

itemMargin(), setItemSpacing(), setMargin(), setSpacing()

12.78.3.29 void QwtPlotLegendItem::setItemSpacing (int spacing)

Set the spacing inside of each item

Parameters

spacing Spacing

See Also

itemSpacing(), setItemMargin(), setMargin(), setSpacing()

12.78.3.30 void QwtPlotLegendItem::setMargin (int margin)

Set the margin around legend items.

The default setting for the margin is 0.

Parameters

margin Margin in pixels

See Also

margin(), setSpacing(), setItemMargin(), setItemSpacing

12.78.3.31 void QwtPlotLegendItem::setMaxColumns (uint maxColumns)

Limit the number of columns.

When aligning the legend horizontally (Qt::AlignLeft, Qt::AlignRight) the number of columns needs to be limited to avoid, that the width of the legend grows with an increasing number of entries.

Parameters

maxColumns | Maximum number of columns. 0 means unlimited.

See Also

maxColumns(), QwtDynGridLayout::setMaxColumns()

12.78.3.32 void QwtPlotLegendItem::setSpacing (int spacing)

Set the spacing between the legend items.

Parameters

spacing | Spacing in pixels

See Also

spacing(), setMargin()

12.78.3.33 void QwtPlotLegendItem::setTextPen (const QPen & pen)

Set the pen for drawing text labels.

Parameters

| | l - . |
|----------|------------------|
| nan | l layt nan |
| Den | I IGAL DOIL |
| <i>I</i> | |

See Also

textPen(), setFont()

12.78.3.34 int QwtPlotLegendItem::spacing () const

Returns

Spacing between the legend items

See Also

setSpacing(), margin(), itemSpacing(), itemMargin()

12.78.3.35 QPen QwtPlotLegendItem::textPen () const

Returns

Pen for drawing text labels

See Also

setTextPen(), font()

12.78.3.36 void QwtPlotLegendItem::updateLegend (const QwtPlotItem * plotItem, const QList< QwtLegendData > & data) [virtual]

Update the legend items according to modifications of a plot item

Parameters

| plotItem | Plot item |
|----------|----------------------------------|
| data | Attributes of the legend entries |

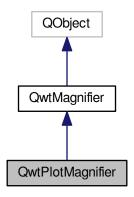
Reimplemented from QwtPlotItem.

12.79 QwtPlotMagnifier Class Reference

QwtPlotMagnifier provides zooming, by magnifying in steps.

#include <qwt_plot_magnifier.h>

Inheritance diagram for QwtPlotMagnifier:



Public Member Functions

- QwtPlotMagnifier (QWidget *)
- virtual ∼QwtPlotMagnifier ()

Destructor.

• void setAxisEnabled (int axis, bool on)

En/Disable an axis.

- bool isAxisEnabled (int axis) const
- QWidget * canvas ()

Return observed plot canvas.

• const QWidget * canvas () const

Return Observed plot canvas.

QwtPlot * plot ()

Return plot widget, containing the observed plot canvas.

const QwtPlot * plot () const

Return plot widget, containing the observed plot canvas.

Protected Member Functions

• virtual void rescale (double factor)

12.79.1 Detailed Description

QwtPlotMagnifier provides zooming, by magnifying in steps.

Using QwtPlotMagnifier a plot can be zoomed in/out in steps using keys, the mouse wheel or moving a mouse button in vertical direction.

Together with QwtPlotZoomer and QwtPlotPanner it is possible to implement individual and powerful navigation of the plot canvas.

See Also

QwtPlotZoomer, QwtPlotPanner, QwtPlot

12.79.2 Constructor & Destructor Documentation

12.79.2.1 QwtPlotMagnifier::QwtPlotMagnifier (QWidget * canvas) [explicit]

Constructor

Parameters

| canvas Plot canvas to be magnified | canvas |
|--------------------------------------|--------|
|--------------------------------------|--------|

12.79.3 Member Function Documentation

12.79.3.1 bool QwtPlotMagnifier::isAxisEnabled (int axis) const

Test if an axis is enabled

Parameters

| axis Axis, see QwtPlot::Axis |
|------------------------------|
|------------------------------|

Returns

True, if the axis is enabled

See Also

setAxisEnabled()

12.79.3.2 void QwtPlotMagnifier::rescale (double *factor* **)** [protected], [virtual]

Zoom in/out the axes scales

Parameters

| factor | A value < 1.0 zooms in, a value > 1.0 zooms out. |
|--------|--|

Implements QwtMagnifier.

12.79.3.3 void QwtPlotMagnifier::setAxisEnabled (int axis, bool on)

En/Disable an axis.

Only Axes that are enabled will be zoomed. All other axes will remain unchanged.

Parameters

| axis | Axis, see QwtPlot::Axis |
|------|-------------------------|
| on | On/Off |

See Also

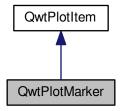
isAxisEnabled()

12.80 QwtPlotMarker Class Reference

A class for drawing markers.

#include <qwt_plot_marker.h>

Inheritance diagram for QwtPlotMarker:



Public Types

• enum LineStyle { NoLine, HLine, VLine, Cross }

Public Member Functions

QwtPlotMarker (const QString &title=QString::null)

Sets alignment to Qt::AlignCenter, and style to QwtPlotMarker::NoLine.

QwtPlotMarker (const QwtText &title)

Sets alignment to Qt::AlignCenter, and style to QwtPlotMarker::NoLine.

• virtual \sim QwtPlotMarker ()

Destructor.

- · virtual int rtti () const
- double xValue () const

Return x Value.

• double yValue () const

Return y Value.

• QPointF value () const

Return Value.

void setXValue (double)

Set X Value.

• void setYValue (double)

Set Y Value.

• void setValue (double, double)

Set Value.

• void setValue (const QPointF &)

Set Value.

• void setLineStyle (LineStyle st)

Set the line style.

- LineStyle lineStyle () const
- void setLinePen (const QColor &, qreal width=0.0, Qt::PenStyle=Qt::SolidLine)
- void setLinePen (const QPen &p)
- const QPen & linePen () const
- void setSymbol (const QwtSymbol *)

Assign a symbol.

• const QwtSymbol * symbol () const

void setLabel (const QwtText &)

Set the label.

- QwtText label () const
- void setLabelAlignment (Qt::Alignment)

Set the alignment of the label.

- Qt::Alignment labelAlignment () const
- void setLabelOrientation (Qt::Orientation)

Set the orientation of the label.

- · Qt::Orientation labelOrientation () const
- void setSpacing (int)

Set the spacing.

- int spacing () const
- virtual void draw (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &) const
- · virtual QRectF boundingRect () const
- virtual QwtGraphic legendlcon (int index, const QSizeF &) const

Protected Member Functions

- virtual void drawLines (QPainter *, const QRectF &, const QPointF &) const
- virtual void drawLabel (QPainter *, const QRectF &, const QPointF &) const

12.80.1 Detailed Description

A class for drawing markers.

A marker can be a horizontal line, a vertical line, a symbol, a label or any combination of them, which can be drawn around a center point inside a bounding rectangle.

The setSymbol() member assigns a symbol to the marker. The symbol is drawn at the specified point.

With setLabel(), a label can be assigned to the marker. The setLabelAlignment() member specifies where the label is drawn. All the Align*-constants in Qt::AlignmentFlags (see Qt documentation) are valid. The interpretation of the alignment depends on the marker's line style. The alignment refers to the center point of the marker, which means, for example, that the label would be printed left above the center point if the alignment was set to Qt::AlignLeft | Qt::AlignTop.

Note

QwtPlotTextLabel is intended to align a text label according to the geometry of canvas (unrelated to plot coordinates)

12.80.2 Member Enumeration Documentation

12.80.2.1 enum QwtPlotMarker::LineStyle

Line styles.

See Also

setLineStyle(), lineStyle()

Enumerator

NoLine No line.

HLine A horizontal line.

VLine A vertical line.

Cross A crosshair.

12.80.3 Member Function Documentation

12.80.3.1 QRectF QwtPlotMarker::boundingRect() const [virtual]

Returns

An invalid bounding rect: QRectF(1.0, 1.0, -2.0, -2.0)

Note

A width or height < 0.0 is ignored by the autoscaler

Reimplemented from QwtPlotItem.

12.80.3.2 void QwtPlotMarker::draw (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect) const [virtual]

Draw the marker

Parameters

| painter | Painter |
|------------|---|
| хМар | x Scale Map |
| уМар | y Scale Map |
| canvasRect | Contents rectangle of the canvas in painter coordinates |

Implements QwtPlotItem.

12.80.3.3 void QwtPlotMarker::drawLabel (QPainter * painter, const QRectF & canvasRect, const QPointF & pos) const [protected], [virtual]

Align and draw the text label of the marker

Parameters

| painter | Painter |
|------------|--|
| canvasRect | Contents rectangle of the canvas in painter coordinates |
| pos | Position of the marker, translated into widget coordinates |

See Also

 $drawLabel(),\ QwtSymbol::drawSymbol()$

12.80.3.4 void QwtPlotMarker::drawLines (QPainter * painter, const QRectF & canvasRect, const QPointF & pos) const [protected], [virtual]

Draw the lines marker

Parameters

| painter | Painter |
|------------|--|
| canvasRect | Contents rectangle of the canvas in painter coordinates |
| pos | Position of the marker, translated into widget coordinates |

See Also

drawLabel(), QwtSymbol::drawSymbol()

12.80.3.5 QwtText QwtPlotMarker::label () const

```
Returns
      the label
See Also
      setLabel()
12.80.3.6 Qt::Alignment QwtPlotMarker::labelAlignment ( ) const
Returns
      the label alignment
See Also
      setLabelAlignment(), setLabelOrientation()
12.80.3.7 Qt::Orientation QwtPlotMarker::labelOrientation ( ) const
Returns
      the label orientation
See Also
      setLabelOrientation(), labelAlignment()
12.80.3.8 QwtGraphic QwtPlotMarker::legendlcon (int index, const QSizeF & size ) const [virtual]
Returns
      Icon representing the marker on the legend
Parameters
                      Index of the legend entry ( usually there is only one )
              index
               size
                      Icon size
```

See Also

```
setLegendIconSize(), legendData()
```

Reimplemented from QwtPlotItem.

12.80.3.9 const QPen & QwtPlotMarker::linePen () const

Returns

the line pen

See Also

setLinePen()

12.80.3.10 QwtPlotMarker::LineStyle QwtPlotMarker::lineStyle () const

Returns

the line style

See Also

setLineStyle()

12.80.3.11 int QwtPlotMarker::rtti() const [virtual]

Returns

QwtPlotItem::Rtti_PlotMarker

Reimplemented from QwtPlotItem.

12.80.3.12 void QwtPlotMarker::setLabel (const QwtText & label)

Set the label.

Parameters

| label | Label text |
|-------|------------|

See Also

label()

12.80.3.13 void QwtPlotMarker::setLabelAlignment (Qt::Alignment align)

Set the alignment of the label.

In case of QwtPlotMarker::HLine the alignment is relative to the y position of the marker, but the horizontal flags correspond to the canvas rectangle. In case of QwtPlotMarker::VLine the alignment is relative to the x position of the marker, but the vertical flags correspond to the canvas rectangle.

In all other styles the alignment is relative to the marker's position.

Parameters

| align | Alignment. |
|-------|------------|
|-------|------------|

See Also

labelAlignment(), labelOrientation()

12.80.3.14 void QwtPlotMarker::setLabelOrientation (Qt::Orientation orientation)

Set the orientation of the label.

When orientation is Qt::Vertical the label is rotated by 90.0 degrees (from bottom to top).

Parameters

```
orientation  Orientation of the label
```

See Also

labelOrientation(), setLabelAlignment()

```
12.80.3.15 void QwtPlotMarker::setLinePen ( const QColor & color, qreal width = 0.0, Qt::PenStyle style = Qt::SolidLine)
```

Build and assign a line pen

In Qt5 the default pen width is 1.0 (0.0 in Qt4) what makes it non cosmetic (see QPen::isCosmetic()). This method has been introduced to hide this incompatibility.

Parameters

| color | Pen color |
|-------|-----------|
| width | Pen width |
| style | Pen style |

See Also

pen(), brush()

12.80.3.16 void QwtPlotMarker::setLinePen (const QPen & pen)

Specify a pen for the line.

Parameters

| pen | New pen |
|-----|---------|
|-----|---------|

See Also

linePen()

12.80.3.17 void QwtPlotMarker::setLineStyle (LineStyle style)

Set the line style.

Parameters

| STVIE | l line style |
|-------|--------------|
| 01,70 | Line of the |

See Also

lineStyle()

12.80.3.18 void QwtPlotMarker::setSpacing (int spacing)

Set the spacing.

When the label is not centered on the marker position, the spacing is the distance between the position and the label.

Parameters

| spacing |
|---------|
|---------|

See Also

spacing(), setLabelAlignment()

12.80.3.19 void QwtPlotMarker::setSymbol (const QwtSymbol * symbol)

Assign a symbol.

Parameters

| symbol | New symbol |
|--------|------------|

See Also

symbol()

```
12.80.3.20 int QwtPlotMarker::spacing ( ) const

Returns
    the spacing

See Also
    setSpacing()

12.80.3.21 const QwtSymbol * QwtPlotMarker::symbol ( ) const

Returns
    the symbol

See Also
```

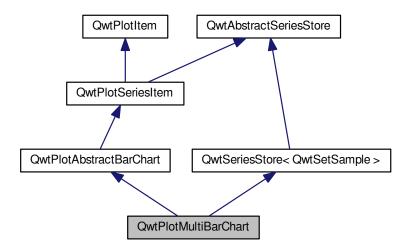
12.81 QwtPlotMultiBarChart Class Reference

setSymbol(), QwtSymbol

QwtPlotMultiBarChart displays a series of a samples that consist each of a set of values.

```
#include <qwt_plot_multi_barchart.h>
```

Inheritance diagram for QwtPlotMultiBarChart:



Public Types

enum ChartStyle { Grouped, Stacked }
 Chart styles.

Public Member Functions

• QwtPlotMultiBarChart (const QString &title=QString::null)

- QwtPlotMultiBarChart (const QwtText &title)
- virtual ~QwtPlotMultiBarChart ()

Destructor.

- virtual int rtti () const
- void setBarTitles (const QList< QwtText > &)

Set the titles for the bars.

- QList< QwtText > barTitles () const
- void setSamples (const QVector< QwtSetSample > &)
- void setSamples (const QVector< QVector< double >> &)
- void setSamples (QwtSeriesData< QwtSetSample > *)
- void setStyle (ChartStyle style)
- · ChartStyle style () const
- void setSymbol (int barIndex, QwtColumnSymbol *symbol)

Add a symbol to the symbol map.

- const QwtColumnSymbol * symbol (int barIndex) const
- void resetSymbolMap ()
- virtual void drawSeries (QPainter *painter, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int from, int to) const
- virtual QRectF boundingRect () const
- virtual QList< QwtLegendData > legendData () const
- virtual QwtGraphic legendlcon (int index, const QSizeF &) const

Protected Member Functions

- QwtColumnSymbol * symbol (int barIndex)
- virtual QwtColumnSymbol * specialSymbol (int sampleIndex, int valueIndex) const

Create a symbol for special values.

- virtual void drawSample (QPainter *painter, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, const QwtInterval &boundingInterval, int index, const QwtSetSample &sample) const
- virtual void drawBar (QPainter *, int sampleIndex, int barIndex, const QwtColumnRect &) const
- void drawStackedBars (QPainter *painter, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int index, double sampleWidth, const QwtSetSample &sample) const
- void drawGroupedBars (QPainter *painter, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int index, double sampleWidth, const QwtSetSample &sample) const

12.81.1 Detailed Description

QwtPlotMultiBarChart displays a series of a samples that consist each of a set of values.

Each value is displayed as a bar, the bars of each set can be organized side by side or accumulated.

Each bar of a set is rendered by a QwtColumnSymbol, that is set by setSymbol(). The bars of different sets use the same symbols. Exceptions are possible by overloading specialSymbol() or overloading drawBar().

Depending on its orientation() the bars are displayed horizontally or vertically. The bars cover the interval between the baseline() and the value.

In opposite to most other plot items, QwtPlotMultiBarChart returns more than one entry for the legend - one for each symbol.

See Also

QwtPlotBarChart, QwtPlotHistogram QwtPlotSeriesItem::orientation(), QwtPlotAbstractBarChart::baseline()

12.81.2 Member Enumeration Documentation

12.81.2.1 enum QwtPlotMultiBarChart::ChartStyle

Chart styles.

The default setting is QwtPlotMultiBarChart::Grouped.

See Also

```
setStyle(), style()
```

Enumerator

Grouped The bars of a set are displayed side by side.

Stacked The bars are displayed on top of each other accumulating to a single bar. All values of a set need to have the same sign.

12.81.3 Constructor & Destructor Documentation

12.81.3.1 QwtPlotMultiBarChart::QwtPlotMultiBarChart (const QString & title = QString::null) [explicit]

Constructor

Parameters

| title Title of the chart | title | |
|----------------------------|-------|--|
|----------------------------|-------|--|

12.81.3.2 QwtPlotMultiBarChart::QwtPlotMultiBarChart (const QwtText & title) [explicit]

Constructor

Parameters

```
title Title of the chart
```

12.81.4 Member Function Documentation

12.81.4.1 QList< QwtText > QwtPlotMultiBarChart::barTitles () const

Returns

Bar titles

See Also

```
setBarTitles(), legendData()
```

12.81.4.2 QRectF QwtPlotMultiBarChart::boundingRect() const [virtual]

Returns

Bounding rectangle of all samples. For an empty series the rectangle is invalid.

Reimplemented from QwtPlotSeriesItem.

12.81.4.3 void QwtPlotMultiBarChart::drawBar (QPainter * painter, int sampleIndex, int valueIndex, const QwtColumnRect & rect) const [protected], [virtual]

Draw a bar

Parameters

| painter | Painter |
|-------------|--|
| sampleIndex | Index of the sample - might be -1 when the bar is painted for the legend |
| valueIndex | Index of a value in a set |
| rect | Directed target rectangle for the bar |

See Also

drawSeries()

12.81.4.4 void QwtPlotMultiBarChart::drawGroupedBars (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int index, double sampleWidth, const QwtSetSample & sample) const [protected]

Draw a grouped sample

Parameters

| painter | Painter |
|-------------|--|
| хМар | x map |
| уМар | y map |
| canvasRect | Contents rectangle of the canvas |
| index | Index of the sample to be painted |
| sampleWidth | Boundng width for all bars of the smaple |
| sample | Sample |

See Also

drawSeries(), sampleWidth()

12.81.4.5 void QwtPlotMultiBarChart::drawSample (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, const QwtInterval & boundingInterval, int index, const QwtSetSample & sample) const [protected], [virtual]

Draw a sample

Parameters

| painter | Painter |
|------------------|------------------------------------|
| хМар | x map |
| уМар | y map |
| canvasRect | Contents rectangle of the canvas |
| boundingInterval | Bounding interval of sample values |
| index | Index of the sample to be painted |
| sample | Sample value |

See Also

drawSeries()

12.81.4.6 void QwtPlotMultiBarChart::drawSeries (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [virtual]

Draw an interval of the bar chart

Parameters

| painter | Painter |
|------------|---|
| хМар | Maps x-values into pixel coordinates. |
| уМар | Maps y-values into pixel coordinates. |
| canvasRect | Contents rectangle of the canvas |
| from | Index of the first point to be painted |
| to | Index of the last point to be painted. If to $<$ 0 the curve will be painted to its last point. |

See Also

drawSymbols()

Implements QwtPlotSeriesItem.

12.81.4.7 void QwtPlotMultiBarChart::drawStackedBars (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int index, double sampleWidth, const QwtSetSample & sample) const [protected]

Draw a stacked sample

Parameters

| painter | Painter |
|-------------|-----------------------------------|
| хМар | x map |
| уМар | y map |
| canvasRect | Contents rectangle of the canvas |
| index | Index of the sample to be painted |
| sampleWidth | Width of the bars |
| sample | Sample |

See Also

drawSeries(), sampleWidth()

 $\textbf{12.81.4.8} \quad \textbf{QList} < \textbf{QwtLegendData} > \textbf{QwtPlotMultiBarChart::legendData() const} \quad \texttt{[virtual]}$

Returns

Information to be displayed on the legend

The chart is represented by a list of entries - one for each bar title. Each element contains a bar title and an icon showing its corresponding bar.

See Also

barTitles(), legendlcon(), legendlconSize()

Reimplemented from QwtPlotItem.

12.81.4.9 QwtGraphic QwtPlotMultiBarChart::legendlcon(int index, const QSizeF & size)const [virtual]

Returns

Icon for representing a bar on the legend

Parameters

| index | Index of the bar |
|-------|------------------|
| size | Icon size |

Returns

An icon showing a bar

See Also

drawBar(), legendData()

Reimplemented from QwtPlotItem.

12.81.4.10 void QwtPlotMultiBarChart::resetSymbolMap()

Remove all symbols from the symbol map

12.81.4.11 int QwtPlotMultiBarChart::rtti() const [virtual]

Returns

QwtPlotItem::Rtti_PlotBarChart

Reimplemented from QwtPlotItem.

12.81.4.12 void QwtPlotMultiBarChart::setBarTitles (const QList< QwtText > & titles)

Set the titles for the bars.

The titles are used for the legend.

Parameters

| titles | Bar titles |
|--------|------------|

See Also

barTitles(), legendData()

12.81.4.13 void QwtPlotMultiBarChart::setSamples (const QVector< QwtSetSample > & samples)

Initialize data with an array of samples.

Parameters

| samples | Vector of points |
|---------|------------------|

12.81.4.14 void QwtPlotMultiBarChart::setSamples (const QVector< QVector< double >> & samples)

Initialize data with an array of samples.

Parameters

samples Vector of points

12.81.4.15 void QwtPlotMultiBarChart::setSamples (QwtSeriesData < QwtSetSample > * data)

Assign a series of samples

setSamples() is just a wrapper for setData() without any additional value - beside that it is easier to find for the developer.

Parameters

| data | Data |
|------|------|
|------|------|

Warning

The item takes ownership of the data object, deleting it when its not used anymore.

12.81.4.16 void QwtPlotMultiBarChart::setStyle (ChartStyle style)

Set the style of the chart

Parameters

| style | Chart style |
|-------|-------------|
|-------|-------------|

See Also

style()

12.81.4.17 void QwtPlotMultiBarChart::setSymbol (int valueIndex, QwtColumnSymbol * symbol)

Add a symbol to the symbol map.

Assign a default symbol for drawing the bar representing all values with the same index in a set.

Parameters

| valueIndex | Index of a value in a set |
|------------|-------------------------------|
| symbol | Symbol used for drawing a bar |

See Also

symbol(), resetSymbolMap(), specialSymbol()

12.81.4.18 QwtColumnSymbol * QwtPlotMultiBarChart::specialSymbol (int sampleIndex, int valueIndex) const [protected], [virtual]

Create a symbol for special values.

Usually the symbols for displaying a bar are set by setSymbols() and common for all sets. By overloading special-Symbol() it is possible to create a temporary symbol() for displaying a special value.

The symbol has to be created by new each time specialSymbol() is called. As soon as the symbol is painted this symbol gets deleted.

When no symbol (NULL) is returned, the value will be displayed with the standard symbol that is used for all symbols with the same valueIndex.

Parameters

| sampleIndex | Index of the sample |
|-------------|-------------------------------|
| valueIndex | Index of the value in the set |

Returns

NULL, meaning that the value is not special

12.81.4.19 QwtPlotMultiBarChart::ChartStyle QwtPlotMultiBarChart::style () const

Returns

Style of the chart

See Also

setStyle()

 $12.81.4.20 \quad const \ \textbf{QwtColumnSymbol} * \ \textbf{QwtPlotMultiBarChart::symbol} \ (\ int \ \textit{valueIndex} \) \ const$

Find a symbol in the symbol map

Parameters

valueIndex Index of a value in a set

Returns

The symbol, that had been set by setSymbol() or NULL.

See Also

setSymbol(), specialSymbol(), drawBar()

12.81.4.21 QwtColumnSymbol * **QwtPlotMultiBarChart::symbol** (int *valueIndex*) [protected]

Find a symbol in the symbol map

Parameters

valueIndex Index of a value in a set

Returns

The symbol, that had been set by setSymbol() or NULL.

See Also

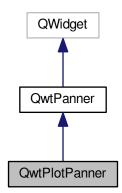
setSymbol(), specialSymbol(), drawBar()

12.82 QwtPlotPanner Class Reference

QwtPlotPanner provides panning of a plot canvas.

#include <qwt_plot_panner.h>

Inheritance diagram for QwtPlotPanner:



Public Member Functions

QwtPlotPanner (QWidget *)

A panner for the canvas of a QwtPlot.

virtual ∼QwtPlotPanner ()

Destructor.

• QWidget * canvas ()

Return observed plot canvas.

• const QWidget * canvas () const

Return Observed plot canvas.

QwtPlot * plot ()

Return plot widget, containing the observed plot canvas.

const QwtPlot * plot () const

Return plot widget, containing the observed plot canvas.

• void setAxisEnabled (int axis, bool on)

En/Disable an axis.

• bool isAxisEnabled (int axis) const

Protected Slots

virtual void moveCanvas (int dx, int dy)

Protected Member Functions

- virtual QBitmap contentsMask () const
- · virtual QPixmap grab () const

Additional Inherited Members

12.82.1 Detailed Description

QwtPlotPanner provides panning of a plot canvas.

QwtPlotPanner is a panner for a plot canvas, that adjusts the scales of the axes after dropping the canvas on its new position.

Together with QwtPlotZoomer and QwtPlotMagnifier powerful ways of navigating on a QwtPlot widget can be implemented easily.

Note

The axes are not updated, while dragging the canvas

See Also

QwtPlotZoomer, QwtPlotMagnifier

12.82.2 Constructor & Destructor Documentation

12.82.2.1 QwtPlotPanner::QwtPlotPanner(QWidget * canvas) [explicit]

A panner for the canvas of a QwtPlot.

The panner is enabled for all axes

Parameters

canvas Plot canvas to pan, also the parent object

See Also

setAxisEnabled()

12.82.3 Member Function Documentation

12.82.3.1 QBitmap QwtPlotPanner::contentsMask() const [protected], [virtual]

Calculate a mask from the border path of the canvas

Returns

Mask as bitmap

See Also

QwtPlotCanvas::borderPath()

Reimplemented from QwtPanner.

12.82.3.2 QPixmap QwtPlotPanner::grab () const [protected], [virtual]

Returns

Pixmap with the content of the canvas

Reimplemented from QwtPanner.

12.82.3.3 bool QwtPlotPanner::isAxisEnabled (int axis) const

Test if an axis is enabled

Parameters

| axis | Axis, see QwtPlot::Axis |
|------|-------------------------|
| | -, |

Returns

True, if the axis is enabled

See Also

setAxisEnabled(), moveCanvas()

12.82.3.4 void QwtPlotPanner::moveCanvas (int dx, int dy) [protected], [virtual], [slot]

Adjust the enabled axes according to dx/dy

Parameters

| dx | Pixel offset in x direction |
|----|-----------------------------|
| dy | Pixel offset in y direction |

See Also

QwtPanner::panned()

12.82.3.5 void QwtPlotPanner::setAxisEnabled (int axis, bool on)

En/Disable an axis.

Axes that are enabled will be synchronized to the result of panning. All other axes will remain unchanged.

Parameters

| axis | Axis, see QwtPlot::Axis |
|------|-------------------------|
| on | On/Off |

See Also

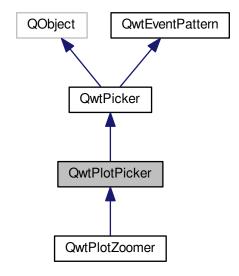
isAxisEnabled(), moveCanvas()

12.83 QwtPlotPicker Class Reference

QwtPlotPicker provides selections on a plot canvas.

#include <qwt_plot_picker.h>

Inheritance diagram for QwtPlotPicker:



Signals

- void selected (const QPointF &pos)
- void selected (const QRectF &rect)
- void selected (const QVector< QPointF > &pa)
- void appended (const QPointF &pos)
- void moved (const QPointF &pos)

Public Member Functions

• QwtPlotPicker (QWidget *canvas)

Create a plot picker.

virtual ~QwtPlotPicker ()

Destructor.

- QwtPlotPicker (int xAxis, int yAxis, QWidget *)
- QwtPlotPicker (int xAxis, int yAxis, RubberBand rubberBand, DisplayMode trackerMode, QWidget *)
- virtual void setAxis (int xAxis, int yAxis)
- int xAxis () const

Return x axis.

• int yAxis () const

Return y axis.

- QwtPlot * plot ()
- const QwtPlot * plot () const
- QWidget * canvas ()
- const QWidget * canvas () const

Protected Member Functions

- QRectF scaleRect () const
- QRectF invTransform (const QRect &) const
- · QRect transform (const QRectF &) const
- QPointF invTransform (const QPoint &) const
- · QPoint transform (const QPointF &) const
- virtual QwtText trackerText (const QPoint &) const
- virtual QwtText trackerTextF (const QPointF &) const

Translate a position into a position string.

- virtual void move (const QPoint &)
- virtual void append (const QPoint &)
- virtual bool end (bool ok=true)

Additional Inherited Members

12.83.1 Detailed Description

QwtPlotPicker provides selections on a plot canvas.

QwtPlotPicker is a QwtPicker tailored for selections on a plot canvas. It is set to a x-Axis and y-Axis and translates all pixel coordinates into this coordinate system.

12.83.2 Constructor & Destructor Documentation

```
12.83.2.1 QwtPlotPicker::QwtPlotPicker( QWidget * canvas ) [explicit]
```

Create a plot picker.

The picker is set to those x- and y-axis of the plot that are enabled. If both or no x-axis are enabled, the picker is set to QwtPlot::xBottom. If both or no y-axis are enabled, it is set to QwtPlot::yLeft.

Parameters

| canvas | Plot canvas to observe, also the parent object |
|--------|--|
|--------|--|

See Also

QwtPlot::autoReplot(), QwtPlot::replot(), scaleRect()

12.83.2.2 QwtPlotPicker::QwtPlotPicker (int xAxis, int yAxis, QWidget * canvas) [explicit]

Create a plot picker

Parameters

| xAxis | Set the x axis of the picker |
|--------|--|
| yAxis | Set the y axis of the picker |
| canvas | Plot canvas to observe, also the parent object |

See Also

QwtPlot::autoReplot(), QwtPlot::replot(), scaleRect()

12.83.2.3 QwtPlotPicker::QwtPlotPicker (int xAxis, int yAxis, RubberBand rubberBand, DisplayMode trackerMode, QWidget * canvas) [explicit]

Create a plot picker

Parameters

| xAxis | X axis of the picker |
|-------------|--|
| yAxis | Y axis of the picker |
| rubberBand | Rubber band style |
| trackerMode | Tracker mode |
| canvas | Plot canvas to observe, also the parent object |

See Also

QwtPicker, QwtPicker::setSelectionFlags(), QwtPicker::setRubberBand(), QwtPicker::setTrackerMode QwtPlot::autoReplot(), QwtPlot::replot(), scaleRect()

12.83.3 Member Function Documentation

```
\textbf{12.83.3.1} \quad \textbf{void QwtPlotPicker::append ( const QPoint \& \textit{pos} \ )} \quad \texttt{[protected], [virtual]}
```

Append a point to the selection and update rubber band and tracker.

Parameters

| pos | Additional point |
|-----|------------------|

See Also

isActive, begin(), end(), move(), appended()

Note

The appended(const QPoint &), appended(const QDoublePoint &) signals are emitted.

Reimplemented from QwtPicker.

```
12.83.3.2 void QwtPlotPicker::appended ( const QPointF & pos ) [signal]
```

A signal emitted when a point has been appended to the selection

Parameters

| pos Position of the appended point. |
|---------------------------------------|
|---------------------------------------|

See Also

```
append(). moved()
```

```
12.83.3.3 QWidget * QwtPlotPicker::canvas ( )
```

Returns

Observed plot canvas

12.83.3.4 const QWidget * QwtPlotPicker::canvas () const

Returns

Observed plot canvas

12.83.3.5 bool QwtPlotPicker::end (bool ok = true) [protected], [virtual]

Close a selection setting the state to inactive.

Parameters

ok If true, complete the selection and emit selected signals otherwise discard the selection.

Returns

True if the selection has been accepted, false otherwise

Reimplemented from QwtPicker.

Reimplemented in QwtPlotZoomer.

12.83.3.6 QRectF QwtPlotPicker::invTransform (const QRect & rect) const [protected]

Translate a rectangle from pixel into plot coordinates

Returns

Rectangle in plot coordinates

See Also

transform()

12.83.3.7 QPointF QwtPlotPicker::invTransform (const QPoint & pos) const [protected]

Translate a point from pixel into plot coordinates

Returns

Point in plot coordinates

See Also

transform()

12.83.3.8 void QwtPlotPicker::move (const QPoint & pos) [protected], [virtual]

Move the last point of the selection

Parameters

pos New position

See Also

isActive, begin(), end(), append()

Note

The moved(const QPoint &), moved(const QDoublePoint &) signals are emitted.

Reimplemented from QwtPicker.

12.83.3.9 void QwtPlotPicker::moved (const QPointF & pos) [signal]

A signal emitted whenever the last appended point of the selection has been moved.

Parameters

pos Position of the moved last point of the selection.

See Also

move(), appended()

12.83.3.10 QwtPlot * QwtPlotPicker::plot ()

Returns

Plot widget, containing the observed plot canvas

12.83.3.11 const QwtPlot * QwtPlotPicker::plot () const

Returns

Plot widget, containing the observed plot canvas

12.83.3.12 QRectF QwtPlotPicker::scaleRect() const [protected]

Returns

Normalized bounding rectangle of the axes

See Also

QwtPlot::autoReplot(), QwtPlot::replot().

12.83.3.13 void QwtPlotPicker::selected (const QPointF & pos) [signal]

A signal emitted in case of QwtPickerMachine::PointSelection.

Parameters

pos | Selected point

12.83.3.14 void QwtPlotPicker::selected (const QRectF & rect) [signal]

A signal emitted in case of QwtPickerMachine::RectSelection.

Parameters

rect | Selected rectangle

12.83.3.15 void QwtPlotPicker::selected (const QVector < QPointF > & pa) [signal]

A signal emitting the selected points, at the end of a selection.

Parameters

pa | Selected points

12.83.3.16 void QwtPlotPicker::setAxis (int xAxis, int yAxis) [virtual]

Set the x and y axes of the picker

Parameters

| X | Axis | X axis |
|-----|--------------|--------|
| y e | 4 <i>xis</i> | Y axis |

Reimplemented in QwtPlotZoomer.

12.83.3.17 QwtText QwtPlotPicker::trackerText (const QPoint & pos) const [protected], [virtual]

Translate a pixel position into a position string

Parameters

| pos | Position in pixel coordinates |
|-----|-------------------------------|

Returns

Position string

Reimplemented from QwtPicker.

12.83.3.18 QwtText QwtPlotPicker::trackerTextF(const QPointF & pos) const [protected], [virtual]

Translate a position into a position string.

In case of HLineRubberBand the label is the value of the y position, in case of VLineRubberBand the value of the x position. Otherwise the label contains x and y position separated by a ',' .

The format for the double to string conversion is "%.4f".

Parameters

| pos | Position | | |
|-----|----------|--|--|
| pos | Position | | |

Returns

Position string

12.83.3.19 QRect QwtPlotPicker::transform (const QRectF & rect) const [protected]

Translate a rectangle from plot into pixel coordinates

Returns

Rectangle in pixel coordinates

See Also

invTransform()

12.83.3.20 QPoint QwtPlotPicker::transform (const QPointF & pos) const [protected]

Translate a point from plot into pixel coordinates

Returns

Point in pixel coordinates

See Also

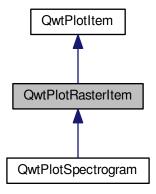
invTransform()

12.84 QwtPlotRasterItem Class Reference

A class, which displays raster data.

#include <qwt_plot_rasteritem.h>

Inheritance diagram for QwtPlotRasterItem:



Public Types

- enum CachePolicy { NoCache, PaintCache }
 - Cache policy The default policy is NoCache.
- enum PaintAttribute { PaintInDeviceResolution = 1 }
- typedef QFlags < PaintAttribute > PaintAttributes

Paint attributes.

Public Member Functions

• QwtPlotRasterItem (const QString &title=QString::null)

Constructor.

QwtPlotRasterItem (const QwtText &title)

Constructor.

virtual ~QwtPlotRasterItem ()

Destructor.

- void setPaintAttribute (PaintAttribute, bool on=true)
- bool testPaintAttribute (PaintAttribute) const
- void setAlpha (int alpha)

Set an alpha value for the raster data.

- int alpha () const
- void setCachePolicy (CachePolicy)
- CachePolicy cachePolicy () const
- void invalidateCache ()
- virtual void draw (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &rect)
 const

Draw the raster data.

virtual QRectF pixelHint (const QRectF &) const

Pixel hint.

- virtual QwtInterval interval (Qt::Axis) const
- · virtual QRectF boundingRect () const

Protected Member Functions

 virtual Qlmage renderlmage (const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &area, const QSize &imageSize) const =0

Render an image.

 virtual QwtScaleMap imageMap (Qt::Orientation, const QwtScaleMap &map, const QRectF &area, const Q-Size &imageSize, double pixelSize) const

Calculate a scale map for painting to an image.

12.84.1 Detailed Description

A class, which displays raster data.

Raster data is a grid of pixel values, that can be represented as a Qlmage. It is used for many types of information like spectrograms, cartograms, geographical maps ...

Often a plot has several types of raster data organized in layers. (f.e a geographical map, with weather statistics). Using setAlpha() raster items can be stacked easily.

QwtPlotRasterItem is only implemented for images of the following formats: Qlmage::Format_Indexed8, Qlmage::Format_ARGB32.

See Also

QwtPlotSpectrogram

12.84.2 Member Enumeration Documentation

12.84.2.1 enum QwtPlotRasterItem::CachePolicy

Cache policy The default policy is NoCache.

Enumerator

NoCache renderImage() is called each time the item has to be repainted

PaintCache renderImage() is called, whenever the image cache is not valid, or the scales, or the size of the canvas has changed.

This type of cache is useful for improving the performance of hide/show operations or manipulations of the alpha value. All other situations are handled by the canvas backing store.

12.84.2.2 enum QwtPlotRasterItem::PaintAttribute

Attributes to modify the drawing algorithm.

See Also

setPaintAttribute(), testPaintAttribute()

Enumerator

PaintInDeviceResolution When the image is rendered according to the data pixels (QwtRasterData::pixel-Hint()) it can be expanded to paint device resolution before it is passed to QPainter. The expansion algorithm rounds the pixel borders in the same way as the axis ticks, what is usually better than the scaling algorithm implemented in Qt. Disabling this flag might make sense, to reduce the size of a document/file. If this is possible for a document format depends on the implementation of the specific QPaintEngine.

12.84.3 Member Function Documentation

12.84.3.1 int QwtPlotRasterItem::alpha () const

Returns

Alpha value of the raster item

See Also

setAlpha()

12.84.3.2 QRectF QwtPlotRasterItem::boundingRect() const [virtual]

Returns

Bounding rectangle of the data

See Also

QwtPlotRasterItem::interval()

Reimplemented from QwtPlotItem.

12.84.3.3 QwtPlotRasterItem::CachePolicy QwtPlotRasterItem::cachePolicy () const

Returns

Cache policy

See Also

CachePolicy, setCachePolicy()

12.84.3.4 void QwtPlotRasterItem::draw (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect) const [virtual]

Draw the raster data.

Parameters

| painter | Painter |
|------------|---------------------------------------|
| хМар | X-Scale Map |
| уМар | Y-Scale Map |
| canvasRect | Contents rectangle of the plot canvas |

Implements QwtPlotItem.

Reimplemented in QwtPlotSpectrogram.

12.84.3.5 QwtScaleMap QwtPlotRasterItem::imageMap (Qt::Orientation *orientation*, const QwtScaleMap & *map*, const QRectF & *area*, const QSize & *imageSize*, double *pixelSize*) const [protected], [virtual]

Calculate a scale map for painting to an image.

Parameters

| orientation | Orientation, Qt::Horizontal means a X axis |
|-------------|--|
| тар | Scale map for rendering the plot item |
| area | Area to be painted on the image |
| imageSize | Image size |
| pixelSize | Width/Height of a data pixel |

Returns

Calculated scale map

12.84.3.6 QwtInterval QwtPlotRasterItem::interval (Qt::Axis axis) const [virtual]

Returns

Bounding interval for an axis

This method is intended to be reimplemented by derived classes. The default implementation returns an invalid interval.

Parameters

| axis | X, Y, or Z axis |
|------|-------------------|
| | 1 1, 1, 1 = 4.115 |

Reimplemented in QwtPlotSpectrogram.

12.84.3.7 void QwtPlotRasterItem::invalidateCache ()

Invalidate the paint cache

See Also

setCachePolicy()

12.84.3.8 QRectF QwtPlotRasterItem::pixelHint (const QRectF & area) const [virtual]

Pixel hint.

The geometry of a pixel is used to calculated the resolution and alignment of the rendered image.

Width and height of the hint need to be the horizontal and vertical distances between 2 neighbored points. The center of the hint has to be the position of any point (it doesn't matter which one).

Limiting the resolution of the image might significantly improve the performance and heavily reduce the amount of memory when rendering a QImage from the raster data.

The default implementation returns an empty rectangle (QRectF()), meaning, that the image will be rendered in target device (f.e screen) resolution.

Parameters

| area | In most implementations the resolution of the data doesn't depend on the requested area. |
|------|--|
|------|--|

Returns

Bounding rectangle of a pixel

See Also

render(), renderImage()

Reimplemented in QwtPlotSpectrogram.

12.84.3.9 virtual Qlmage QwtPlotRasterItem::renderImage (const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & area, const QSize & imageSize) const [protected], [pure virtual]

Render an image.

An implementation of render() might iterate over all pixels of imageRect. Each pixel has to be translated into the corresponding position in scale coordinates using the maps. This position can be used to look up a value in a implementation specific way and to map it into a color.

Parameters

| хМар | X-Scale Map |
|-----------|---|
| уМар | Y-Scale Map |
| area | Requested area for the image in scale coordinates |
| imageSize | Requested size of the image |

Returns

Rendered image

Implemented in QwtPlotSpectrogram.

12.84.3.10 void QwtPlotRasterItem::setAlpha (int alpha)

Set an alpha value for the raster data.

Often a plot has several types of raster data organized in layers. (f.e a geographical map, with weather statistics). Using setAlpha() raster items can be stacked easily.

The alpha value is a value [0, 255] to control the transparency of the image. 0 represents a fully transparent color, while 255 represents a fully opaque color.

Parameters

| aipria Alpria value |
|---------------------|
|---------------------|

• alpha >= 0

All alpha values of the pixels returned by renderImage() will be set to alpha, beside those with an alpha value of 0 (invalid pixels).

• alpha < 0 The alpha values returned by renderImage() are not changed.

The default alpha value is -1.

See Also

alpha()

12.84.3.11 void QwtPlotRasterItem::setCachePolicy (QwtPlotRasterItem::CachePolicy policy)

Change the cache policy

The default policy is NoCache

Parameters

| policy | Cache policy |
|--------|--------------|

See Also

CachePolicy, cachePolicy()

12.84.3.12 void QwtPlotRasterItem::setPaintAttribute (PaintAttribute attribute, bool on = true)

Specify an attribute how to draw the raster item

Parameters

| attribute | Paint attribute |
|-----------|---|
| on | On/Off /sa PaintAttribute, testPaintAttribute() |

12.84.3.13 bool QwtPlotRasterItem::testPaintAttribute (PaintAttribute attribute) const

Returns

True, when attribute is enabled

See Also

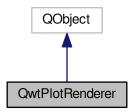
PaintAttribute, setPaintAttribute()

12.85 QwtPlotRenderer Class Reference

Renderer for exporting a plot to a document, a printer or anything else, that is supported by QPainter/QPaintDevice.

#include <qwt_plot_renderer.h>

Inheritance diagram for QwtPlotRenderer:



Public Types

enum DiscardFlag {
 DiscardNone = 0x00, DiscardBackground = 0x01, DiscardTitle = 0x02, DiscardLegend = 0x04,
 DiscardCanvasBackground = 0x08, DiscardFooter = 0x10, DiscardCanvasFrame = 0x20 }

Disard flags

enum LayoutFlag { DefaultLayout = 0x00, FrameWithScales = 0x01 }

Layout flags.

typedef QFlags < DiscardFlag > DiscardFlags

Disard flags.

typedef QFlags < LayoutFlags

Layout flags.

Public Member Functions

- QwtPlotRenderer (QObject *=NULL)
- virtual ~QwtPlotRenderer ()

Destructor.

- void setDiscardFlag (DiscardFlag flag, bool on=true)
- · bool testDiscardFlag (DiscardFlag flag) const
- void setDiscardFlags (DiscardFlags flags)
- · DiscardFlags discardFlags () const
- void setLayoutFlag (LayoutFlag flag, bool on=true)
- · bool testLayoutFlag (LayoutFlag flag) const
- void setLayoutFlags (LayoutFlags flags)
- · LayoutFlags layoutFlags () const
- void renderDocument (QwtPlot *, const QString &fileName, const QSizeF &sizeMM, int resolution=85)
- void renderDocument (QwtPlot *, const QString &fileName, const QString &format, const QSizeF &sizeMM, int resolution=85)
- void renderTo (QwtPlot *, QPrinter &) const

Render the plot to a QPrinter.

void renderTo (QwtPlot *, QPaintDevice &p) const

Render the plot to a QPaintDevice.

- virtual void render (QwtPlot *, QPainter *, const QRectF &rect) const
- virtual void renderTitle (const QwtPlot *, QPainter *, const QRectF &) const
- virtual void renderFooter (const QwtPlot *, QPainter *, const QRectF &) const
- virtual void renderScale (const QwtPlot *, QPainter *, int axisId, int startDist, int endDist, int baseDist, const QRectF &) const

Paint a scale into a given rectangle. Paint the scale into a given rectangle.

- virtual void renderCanvas (const QwtPlot *, QPainter *, const QRectF &canvasRect, const QwtScaleMap *maps) const
- virtual void renderLegend (const QwtPlot *, QPainter *, const QRectF &) const
- bool exportTo (QwtPlot *, const QString &documentName, const QSizeF &sizeMM=QSizeF(300, 200), int resolution=85)

Execute a file dialog and render the plot to the selected file.

12.85.1 Detailed Description

Renderer for exporting a plot to a document, a printer or anything else, that is supported by QPainter/QPaintDevice.

12.85.2 Member Enumeration Documentation

12.85.2.1 enum QwtPlotRenderer::DiscardFlag

Disard flags.

Enumerator

DiscardNone Render all components of the plot.

DiscardBackground Don't render the background of the plot.

DiscardTitle Don't render the title of the plot.

DiscardLegend Don't render the legend of the plot.

DiscardCanvasBackground Don't render the background of the canvas.

DiscardFooter Don't render the footer of the plot.

DiscardCanvasFrame Don't render the frame of the canvas

Note

This flag has no effect when using style sheets, where the frame is part of the background

12.85.2.2 enum QwtPlotRenderer::LayoutFlag

Layout flags.

See Also

setLayoutFlag(), testLayoutFlag()

Enumerator

DefaultLayout Use the default layout as on screen.

FrameWithScales Instead of the scales a box is painted around the plot canvas, where the scale ticks are aligned to.

12.85.3 Constructor & Destructor Documentation

```
12.85.3.1 QwtPlotRenderer::QwtPlotRenderer ( QObject * parent = NULL ) [explicit]
```

Constructor

Parameters

| parent | Parent object |
|--------|---------------|

12.85.4 Member Function Documentation

12.85.4.1 QwtPlotRenderer::DiscardFlags QwtPlotRenderer::discardFlags () const

Returns

Flags, indicating what to discard from rendering

See Also

DiscardFlag, setDiscardFlags(), setDiscardFlag(), testDiscardFlag()

12.85.4.2 bool QwtPlotRenderer::exportTo (QwtPlot * plot, const QString & documentName, const QSizeF & sizeMM = QSizeF (300, 200), int resolution = 85)

Execute a file dialog and render the plot to the selected file.

Parameters

| plot | Plot widget |
|--------------|---------------------------------------|
| documentName | Default document name |
| sizeMM | Size for the document in millimeters. |
| resolution | Resolution in dots per Inch (dpi) |

Returns

True, when exporting was successful

See Also

renderDocument()

12.85.4.3 QwtPlotRenderer::LayoutFlags QwtPlotRenderer::layoutFlags () const

Returns

Layout flags

See Also

LayoutFlag, setLayoutFlags(), setLayoutFlag(), testLayoutFlag()

12.85.4.4 void QwtPlotRenderer::render (QwtPlot * plot, QPainter * painter, const QRectF & plotRect) const [virtual]

Paint the contents of a QwtPlot instance into a given rectangle.

Parameters

| plot | Plot to be rendered |
|----------|---------------------|
| painter | Painter |
| plotRect | Bounding rectangle |

See Also

renderDocument(), renderTo(), QwtPainter::setRoundingAlignment()

12.85.4.5 void QwtPlotRenderer::renderCanvas (const QwtPlot * plot, QPainter * painter, const QRectF & canvasRect, const QwtScaleMap * map) const [virtual]

Render the canvas into a given rectangle.

Parameters

| | plot | Plot widget |
|---|------------|--|
| Ī | painter | Painter |
| Ī | тар | Maps mapping between plot and paint device coordinates |
| Ī | canvasRect | Canvas rectangle |

12.85.4.6 void QwtPlotRenderer::renderDocument (QwtPlot * plot, const QString & fileName, const QSizeF & sizeMM, int resolution = 85)

Render a plot to a file

The format of the document will be auto-detected from the suffix of the file name.

Parameters

| plot | Plot widget |
|------------|---|
| fileName | Path of the file, where the document will be stored |
| sizeMM | Size for the document in millimeters. |
| resolution | Resolution in dots per Inch (dpi) |

12.85.4.7 void QwtPlotRenderer::renderDocument (QwtPlot * plot, const QString & fileName, const QString & format, const QSizeF & sizeMM, int resolution = 85)

Render a plot to a file

Supported formats are:

pdf

Portable Document Format PDF

ps

Postcript

svg

Scalable Vector Graphics SVG

 all image formats supported by Qt see QlmageWriter::supportedImageFormats()

Scalable vector graphic formats like PDF or SVG are superior to raster graphics formats.

Parameters

| plot | Plot widget |
|------------|---|
| fileName | Path of the file, where the document will be stored |
| format | Format for the document |
| sizeMM | Size for the document in millimeters. |
| resolution | Resolution in dots per Inch (dpi) |

See Also

renderTo(), render(), QwtPainter::setRoundingAlignment()

12.85.4.8 void QwtPlotRenderer::renderFooter (const QwtPlot * plot, QPainter * painter, const QRectF & rect) const [virtual]

Render the footer into a given rectangle.

Parameters

| plot | Plot widget |
|---------|--------------------|
| painter | Painter |
| rect | Bounding rectangle |

12.85.4.9 void QwtPlotRenderer::renderLegend (const QwtPlot * plot, QPainter * painter, const QRectF & rect) const [virtual]

Render the legend into a given rectangle.

Parameters

| plot | Plot widget |
|---------|--------------------|
| painter | Painter |
| rect | Bounding rectangle |

12.85.4.10 void QwtPlotRenderer::renderScale (const QwtPlot * plot, QPainter * painter, int axisId, int startDist, int endDist, int baseDist, const QRectF & rect) const [virtual]

Paint a scale into a given rectangle. Paint the scale into a given rectangle.

Parameters

| plot | Plot widget |
|---------|-------------|
| painter | Painter |
| axisId | Axis |

| startDist | Start border distance |
|-----------|-----------------------|
| endDist | End border distance |
| baseDist | Base distance |
| rect | Bounding rectangle |

12.85.4.11 void QwtPlotRenderer::renderTitle (const QwtPlot * plot, QPainter * painter, const QRectF & rect) const [virtual]

Render the title into a given rectangle.

Parameters

| plot | Plot widget |
|---------|--------------------|
| painter | Painter |
| rect | Bounding rectangle |

12.85.4.12 void QwtPlotRenderer::renderTo (QwtPlot * plot, QPrinter & printer) const

Render the plot to a QPrinter.

This function renders the contents of a QwtPlot instance to QPaintDevice object. The size is derived from the printer metrics.

Parameters

| plot | Plot to be rendered |
|---------|---------------------|
| printer | Printer to paint on |

See Also

renderDocument(), render(), QwtPainter::setRoundingAlignment()

12.85.4.13 void QwtPlotRenderer::renderTo (QwtPlot * plot, QPaintDevice & paintDevice) const

Render the plot to a QPaintDevice.

This function renders the contents of a QwtPlot instance to QPaintDevice object. The target rectangle is derived from its device metrics.

Parameters

| plot | Plot to be rendered |
|-------------|----------------------------------|
| paintDevice | device to paint on, f.e a QImage |

See Also

renderDocument(), render(), QwtPainter::setRoundingAlignment()

12.85.4.14 void QwtPlotRenderer::setDiscardFlag (DiscardFlag flag, bool on = true)

Change a flag, indicating what to discard from rendering

Parameters

| flag | Flag to change |
|------|----------------|
| on | On/Off |

See Also

DiscardFlag, testDiscardFlag(), setDiscardFlags(), discardFlags()

12.85.4.15 void QwtPlotRenderer::setDiscardFlags (DiscardFlags flags)

Set the flags, indicating what to discard from rendering

See Also

DiscardFlag, setDiscardFlag(), testDiscardFlag(), discardFlags()

12.85.4.16 void QwtPlotRenderer::setLayoutFlag (LayoutFlag flag, bool on = true)

Change a layout flag

Parameters

| flag | Flag to change |
|------|----------------|
| on | On/Off |

See Also

LayoutFlag, testLayoutFlag(), setLayoutFlags(), layoutFlags()

12.85.4.17 void QwtPlotRenderer::setLayoutFlags (LayoutFlags flags)

Set the layout flags

Parameters

| flags | Flags |
|-------|-------|

See Also

LayoutFlag, setLayoutFlag(), testLayoutFlag(), layoutFlags()

12.85.4.18 bool QwtPlotRenderer::testDiscardFlag (DiscardFlag flag) const

Returns

True, if flag is enabled.

Parameters

| flag | Flag to be tested |
|------|-------------------|

See Also

DiscardFlag, setDiscardFlags(), setDiscardFlags(), discardFlags()

12.85.4.19 bool QwtPlotRenderer::testLayoutFlag (LayoutFlag flag) const

Returns

True, if flag is enabled.

Parameters

| flaa | Flag to be tested |
|------|--------------------|
| iiau | I lay to be tested |

See Also

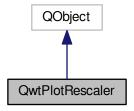
LayoutFlag, setLayoutFlags(), setLayoutFlags(), layoutFlags()

12.86 QwtPlotRescaler Class Reference

QwtPlotRescaler takes care of fixed aspect ratios for plot scales.

```
#include <qwt_plot_rescaler.h>
```

Inheritance diagram for QwtPlotRescaler:



Public Types

- enum RescalePolicy { Fixed, Expanding, Fitting }
- enum ExpandingDirection { ExpandUp, ExpandDown, ExpandBoth }

Public Member Functions

- QwtPlotRescaler (QWidget *canvas, int referenceAxis=QwtPlot::xBottom, RescalePolicy=Expanding)
- virtual ~QwtPlotRescaler ()

Destructor.

void setEnabled (bool)

En/disable the rescaler.

- bool isEnabled () const
- void setRescalePolicy (RescalePolicy)
- · RescalePolicy rescalePolicy () const
- void setExpandingDirection (ExpandingDirection)
- void setExpandingDirection (int axis, ExpandingDirection)
- ExpandingDirection expandingDirection (int axis) const
- void setReferenceAxis (int axis)
- int referenceAxis () const
- void setAspectRatio (double ratio)
- void setAspectRatio (int axis, double ratio)
- double aspectRatio (int axis) const
- void setIntervalHint (int axis, const QwtInterval &)
- · QwtInterval intervalHint (int axis) const
- QWidget * canvas ()

- const QWidget * canvas () const
- QwtPlot * plot ()
- const QwtPlot * plot () const
- virtual bool eventFilter (QObject *, QEvent *)

Event filter for the plot canvas.

· void rescale () const

Adjust the plot axes scales.

Protected Member Functions

- virtual void canvasResizeEvent (QResizeEvent *)
- virtual void rescale (const QSize &oldSize, const QSize &newSize) const
- · virtual QwtInterval expandScale (int axis, const QSize &oldSize, const QSize &newSize) const
- virtual QwtInterval syncScale (int axis, const QwtInterval &reference, const QSize &size) const
- virtual void updateScales (QwtInterval intervals[QwtPlot::axisCnt]) const
- · Qt::Orientation orientation (int axis) const
- · QwtInterval interval (int axis) const
- QwtInterval expandInterval (const QwtInterval &, double width, ExpandingDirection) const

12.86.1 Detailed Description

QwtPlotRescaler takes care of fixed aspect ratios for plot scales.

QwtPlotRescaler auto adjusts the axes of a QwtPlot according to fixed aspect ratios.

12.86.2 Member Enumeration Documentation

12.86.2.1 enum QwtPlotRescaler::ExpandingDirection

When rescalePolicy() is set to Expanding its direction depends on ExpandingDirection

Enumerator

ExpandUp The upper limit of the scale is adjusted.

ExpandDown The lower limit of the scale is adjusted.

ExpandBoth Both limits of the scale are adjusted.

12.86.2.2 enum QwtPlotRescaler::RescalePolicy

The rescale policy defines how to rescale the reference axis and their depending axes.

See Also

ExpandingDirection, setIntervalHint()

Enumerator

Fixed The interval of the reference axis remains unchanged, when the geometry of the canvas changes. All other axes will be adjusted according to their aspect ratio.

Expanding The interval of the reference axis will be shrunk/expanded, when the geometry of the canvas changes. All other axes will be adjusted according to their aspect ratio.

The interval, that is represented by one pixel is fixed.

Fitting The intervals of the axes are calculated, so that all axes include their interval hint.

- 12.86.3 Constructor & Destructor Documentation
- 12.86.3.1 QwtPlotRescaler::QwtPlotRescaler (QWidget * canvas, int referenceAxis = QwtPlot::xBottom, RescalePolicy policy = Expanding) [explicit]

Constructor

| canvas | Canvas |
|---------------|-----------------------------------|
| referenceAxis | Reference axis, see RescalePolicy |
| policy | Rescale policy |

See Also

setRescalePolicy(), setReferenceAxis()

12.86.4 Member Function Documentation

12.86.4.1 double QwtPlotRescaler::aspectRatio (int axis) const

Returns

Aspect ratio between an axis and the reference axis.

Parameters

| axis | Axis index (see QwtPlot::AxisId) |
|------|------------------------------------|

See Also

setAspectRatio()

12.86.4.2 QWidget * QwtPlotRescaler::canvas ()

Returns

plot canvas

12.86.4.3 const QWidget * QwtPlotRescaler::canvas () const

Returns

plot canvas

12.86.4.4 void QwtPlotRescaler::canvasResizeEvent (QResizeEvent * event) [protected], [virtual]

Event handler for resize events of the plot canvas

Parameters

| event | Resize event |
|-------|--------------|

See Also

rescale()

 $12.86.4.5 \quad \textbf{QwtPlotRescaler::} \textbf{ExpandingDirection QwtPlotRescaler::} \textbf{expandingDirection (int} \ \textbf{axis} \ \textbf{)} \ \textbf{const}$

Returns

Direction in which an axis should be expanded

Parameters

| axis | Axis index (see QwtPlot::AxisId) |
|------|------------------------------------|

See Also

setExpandingDirection()

12.86.4.6 QwtInterval QwtPlotRescaler::expandInterval (const QwtInterval & interval, double width, ExpandingDirection direction) const [protected]

Expand the interval

Parameters

| interval | Interval to be expanded |
|-----------|--------------------------------------|
| width | Distance to be added to the interval |
| direction | Direction of the expand operation |

Returns

Expanded interval

12.86.4.7 QwtInterval QwtPlotRescaler::expandScale (int axis, const QSize & oldSize, const QSize & newSize) const [protected], [virtual]

Calculate the new scale interval of a plot axis

Parameters

| axis | Axis index (see QwtPlot::AxisId) |
|---------|------------------------------------|
| oldSize | Previous size of the canvas |
| newSize | New size of the canvas |

Returns

Calculated new interval for the axis

12.86.4.8 QwtInterval QwtPlotRescaler::interval (int axis) const [protected]

Parameters

| axis | Axis index (see QwtPlot::AxisId) |
|------|------------------------------------|

Returns

Normalized interval of an axis

12.86.4.9 QwtInterval QwtPlotRescaler::intervalHint (int axis) const

Parameters

| axis | Axis, see QwtPlot::Axis |
|------|-------------------------|

Returns

Interval hint

See Also

setIntervalHint(), RescalePolicy

12.86.4.10 bool QwtPlotRescaler::isEnabled () const

Returns

true when enabled, false otherwise

See Also

setEnabled, eventFilter()

12.86.4.11 Qt::Orientation QwtPlotRescaler::orientation (int axis) const [protected]

Returns

Orientation of an axis

Parameters

```
axis  Axis index ( see QwtPlot::AxisId )
```

12.86.4.12 QwtPlot * QwtPlotRescaler::plot ()

Returns

plot widget

12.86.4.13 const QwtPlot * QwtPlotRescaler::plot () const

Returns

plot widget

12.86.4.14 int QwtPlotRescaler::referenceAxis () const

Returns

Reference axis (see RescalePolicy)

See Also

setReferenceAxis()

12.86.4.15 void QwtPlotRescaler::rescale (const QSize & oldSize, const QSize & newSize) const [protected], [virtual]

Adjust the plot axes scales

Parameters

| oldSize | Previous size of the canvas |
|---------|-----------------------------|
| newSize | New size of the canvas |

12.86.4.16 QwtPlotRescaler::RescalePolicy QwtPlotRescaler::rescalePolicy () const

Returns

Rescale policy

See Also

setRescalePolicy()

12.86.4.17 void QwtPlotRescaler::setAspectRatio (double ratio)

Set the aspect ratio between the scale of the reference axis and the other scales. The default ratio is 1.0

| ratio | Aspect ratio |
|-------|--------------|
|-------|--------------|

See Also

aspectRatio()

12.86.4.18 void QwtPlotRescaler::setAspectRatio (int axis, double ratio)

Set the aspect ratio between the scale of the reference axis and another scale. The default ratio is 1.0

Parameters

| axis | Axis index (see QwtPlot::AxisId) |
|-------|------------------------------------|
| ratio | Aspect ratio |

See Also

aspectRatio()

12.86.4.19 void QwtPlotRescaler::setEnabled (bool on)

En/disable the rescaler.

When enabled is true an event filter is installed for the canvas, otherwise the event filter is removed.

Parameters

| on t | true or false |
|------|---------------|
|------|---------------|

See Also

isEnabled(), eventFilter()

12.86.4.20 void QwtPlotRescaler::setExpandingDirection (ExpandingDirection direction)

Set the direction in which all axis should be expanded

Parameters

| direction | Direction |
|-----------|-----------|
|-----------|-----------|

See Also

expandingDirection()

12.86.4.21 void QwtPlotRescaler::setExpandingDirection (int axis, ExpandingDirection direction)

Set the direction in which an axis should be expanded

Parameters

| axis | Axis index (see QwtPlot::AxisId) |
|-----------|------------------------------------|
| direction | Direction |

See Also

expandingDirection()

12.86.4.22 void QwtPlotRescaler::setIntervalHint (int axis, const QwtInterval & interval)

Set an interval hint for an axis

In Fitting mode, the hint is used as minimal interval that always needs to be displayed.

| axis | Axis, see QwtPlot::Axis |
|----------|-------------------------|
| interval | Axis |

See Also

intervalHint(), RescalePolicy

12.86.4.23 void QwtPlotRescaler::setReferenceAxis (int axis)

Set the reference axis (see RescalePolicy)

Parameters

| axis | Axis index (QwtPlot::Axis) |
|------|------------------------------|
|------|------------------------------|

See Also

referenceAxis()

12.86.4.24 void QwtPlotRescaler::setRescalePolicy (RescalePolicy policy)

Change the rescale policy

Parameters

| policy | Rescale policy |
|--------|----------------|

See Also

rescalePolicy()

12.86.4.25 QwtInterval QwtPlotRescaler::syncScale (int axis, const QwtInterval & reference, const QSize & size) const [protected], [virtual]

Synchronize an axis scale according to the scale of the reference axis

Parameters

| axis | Axis index (see QwtPlot::AxisId) |
|-----------|------------------------------------|
| reference | Interval of the reference axis |
| size | Size of the canvas |

Returns

New interval for axis

12.86.4.26 void QwtPlotRescaler::updateScales (QwtInterval *intervals[QwtPlot::axisCnt]* **) const** [protected], [virtual]

Update the axes scales

Parameters

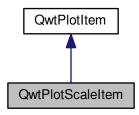
| intervals | Scale intervals |
|-----------|-----------------|

12.87 QwtPlotScaleItem Class Reference

A class which draws a scale inside the plot canvas.

#include <qwt_plot_scaleitem.h>

Inheritance diagram for QwtPlotScaleItem:



Public Member Functions

 $\bullet \ \ QwtPlotScaleItem \ (QwtScaleDraw::Alignment=QwtScaleDraw::BottomScale, \ const \ double \ pos=0.0)$

Constructor for scale item at the position pos.

virtual ~QwtPlotScaleItem ()

Destructor.

- · virtual int rtti () const
- void setScaleDiv (const QwtScaleDiv &)

Assign a scale division.

- const QwtScaleDiv & scaleDiv () const
- void setScaleDivFromAxis (bool on)
- · bool isScaleDivFromAxis () const
- void setPalette (const QPalette &)
- QPalette palette () const
- void setFont (const QFont &)
- · QFont font () const
- void setScaleDraw (QwtScaleDraw *)

Set a scale draw.

- const QwtScaleDraw * scaleDraw () const
- QwtScaleDraw * scaleDraw ()
- void setPosition (double pos)
- double position () const
- void setBorderDistance (int numPixels)

Align the scale to the canvas.

- int borderDistance () const
- void setAlignment (QwtScaleDraw::Alignment)
- virtual void draw (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &rect)
 const

Draw the scale.

virtual void updateScaleDiv (const QwtScaleDiv &, const QwtScaleDiv &)

Update the item to changes of the axes scale division.

Additional Inherited Members

12.87.1 Detailed Description

A class which draws a scale inside the plot canvas.

QwtPlotScaleItem can be used to draw an axis inside the plot canvas. It might by synchronized to one of the axis of the plot, but can also display its own ticks and labels.

It is allowed to synchronize the scale item with a disabled axis. In plots with vertical and horizontal scale items, it might be necessary to remove ticks at the intersections, by overloading updateScaleDiv().

The scale might be at a specific position (f.e 0.0) or it might be aligned to a canvas border.

Example

The following example shows how to replace the left axis, by a scale item at the x position 0.0.

```
QwtPlotScaleItem *scaleItem =
    new QwtPlotScaleItem(QwtScaleDraw::RightScale, 0.0);
scaleItem->setFont(plot->axisWidget(QwtPlot::yLeft)->font());
scaleItem->attach(plot);
plot->enableAxis(QwtPlot::yLeft, false);
```

12.87.2 Constructor & Destructor Documentation

```
12.87.2.1 QwtPlotScaleItem::QwtPlotScaleItem ( QwtScaleDraw::Alignment alignment = QwtScaleDraw::BottomScale, const double pos = 0.0) [explicit]
```

Constructor for scale item at the position pos.

Parameters

| alignment | In case of QwtScaleDraw::BottomScale or QwtScaleDraw::TopScale the scale item is corre- |
|-----------|---|
| | sponding to the xAxis(), otherwise it corresponds to the yAxis(). |
| pos | x or y position, depending on the corresponding axis. |

See Also

```
setPosition(), setAlignment()
```

12.87.3 Member Function Documentation

12.87.3.1 int QwtPlotScaleItem::borderDistance () const

Returns

Distance from a canvas border

See Also

```
setBorderDistance(), setPosition()
```

```
12.87.3.2 QFont QwtPlotScaleItem::font ( ) const
```

Returns

tick label font

See Also

setFont()

```
12.87.3.3 bool QwtPlotScaleItem::isScaleDivFromAxis ( ) const
Returns
      True, if the synchronization of the scale division with the corresponding axis is enabled.
See Also
      setScaleDiv(), setScaleDivFromAxis()
12.87.3.4 QPalette QwtPlotScaleItem::palette ( ) const
Returns
     palette
See Also
     setPalette()
12.87.3.5 double QwtPlotScaleItem::position ( ) const
Returns
      Position of the scale
See Also
      setPosition(), setAlignment()
12.87.3.6 int QwtPlotScaleItem::rtti() const [virtual]
Returns
      QwtPlotItem::Rtti_PlotScale
Reimplemented from QwtPlotItem.
12.87.3.7 const QwtScaleDiv & QwtPlotScaleItem::scaleDiv ( ) const
Returns
      Scale division
12.87.3.8 const QwtScaleDraw * QwtPlotScaleItem::scaleDraw ( ) const
Returns
      Scale draw
See Also
      setScaleDraw()
12.87.3.9 QwtScaleDraw * QwtPlotScaleItem::scaleDraw ( )
Returns
      Scale draw
See Also
      setScaleDraw()
```

12.87.3.10 void QwtPlotScaleItem::setAlignment (QwtScaleDraw::Alignment alignment)

Change the alignment of the scale

The alignment sets the orientation of the scale and the position of the ticks:

QwtScaleDraw::BottomScale: horizontal, ticks below

• QwtScaleDraw::TopScale: horizontal, ticks above

· QwtScaleDraw::LeftScale: vertical, ticks left

· QwtScaleDraw::RightScale: vertical, ticks right

For horizontal scales the position corresponds to QwtPlotItem::yAxis(), otherwise to QwtPlotItem::xAxis().

See Also

```
scaleDraw(), QwtScaleDraw::alignment(), setPosition()
```

12.87.3.11 void QwtPlotScaleItem::setBorderDistance (int distance)

Align the scale to the canvas.

If distance is >= 0 the scale will be aligned to a border of the contents rectangle of the canvas. If alignment() is QwtScaleDraw::LeftScale, the scale will be aligned to the right border, if it is QwtScaleDraw::TopScale it will be aligned to the bottom (and vice versa),

If distance is < 0 the scale will be at the position().

Parameters

distance | Number of pixels between the canvas border and the backbone of the scale.

See Also

```
setPosition(), borderDistance()
```

12.87.3.12 void QwtPlotScaleItem::setFont (const QFont & font)

Change the tick label font

See Also

font()

12.87.3.13 void QwtPlotScaleItem::setPalette (const QPalette & palette)

Set the palette

See Also

QwtAbstractScaleDraw::draw(), palette()

12.87.3.14 void QwtPlotScaleItem::setPosition (double pos)

Change the position of the scale

The position is interpreted as y value for horizontal axes and as x value for vertical axes.

The border distance is set to -1.

Parameters

pos New position

See Also

position(), setAlignment()

12.87.3.15 void QwtPlotScaleItem::setScaleDiv (const QwtScaleDiv & scaleDiv)

Assign a scale division.

When assigning a scaleDiv the scale division won't be synchronized with the corresponding axis anymore.

Parameters

scaleDiv Scale division

See Also

scaleDiv(), setScaleDivFromAxis(), isScaleDivFromAxis()

12.87.3.16 void QwtPlotScaleItem::setScaleDivFromAxis (bool on)

Enable/Disable the synchronization of the scale division with the corresponding axis.

Parameters

on true/false

See Also

isScaleDivFromAxis()

12.87.3.17 void QwtPlotScaleItem::setScaleDraw (QwtScaleDraw * scaleDraw)

Set a scale draw.

Parameters

scaleDraw object responsible for drawing scales.

The main use case for replacing the default QwtScaleDraw is to overload QwtAbstractScaleDraw::label, to replace or swallow tick labels.

See Also

scaleDraw()

12.87.3.18 void QwtPlotScaleItem::updateScaleDiv (const QwtScaleDiv & xScaleDiv, const QwtScaleDiv & yScaleDiv)
[virtual]

Update the item to changes of the axes scale division.

In case of isScaleDivFromAxis(), the scale draw is synchronized to the correspond axis.

Parameters

| xScaleDiv | Scale division of the x-axis |
|-----------|------------------------------|
| yScaleDiv | Scale division of the y-axis |

See Also

QwtPlot::updateAxes()

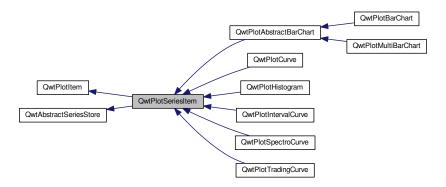
Reimplemented from QwtPlotItem.

12.88 QwtPlotSeriesItem Class Reference

Base class for plot items representing a series of samples.

#include <qwt_plot_seriesitem.h>

Inheritance diagram for QwtPlotSeriesItem:



Public Member Functions

- QwtPlotSeriesItem (const QString &title=QString::null)
- QwtPlotSeriesItem (const QwtText &title)
- virtual ~QwtPlotSeriesItem ()

Destructor.

- void setOrientation (Qt::Orientation)
- Qt::Orientation orientation () const
- virtual void draw (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &) const

Draw the complete series.

- virtual void drawSeries (QPainter *painter, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int from, int to) const =0
- · virtual QRectF boundingRect () const
- virtual void updateScaleDiv (const QwtScaleDiv &, const QwtScaleDiv &)

Update the item to changes of the axes scale division.

Protected Member Functions

virtual void dataChanged ()

dataChanged() indicates, that the series has been changed.

Additional Inherited Members

12.88.1 Detailed Description

Base class for plot items representing a series of samples.

12.88.2 Constructor & Destructor Documentation

12.88.2.1 QwtPlotSeriesItem::QwtPlotSeriesItem (const QString & title = QString::null) [explicit]

Constructor

Parameters

| title | Title of the curve |
|-------|--------------------|

12.88.2.2 QwtPlotSeriesItem::QwtPlotSeriesItem (const QwtText & title) [explicit]

Constructor

Parameters

| title Title of the curve |
|--------------------------|
|--------------------------|

12.88.3 Member Function Documentation

12.88.3.1 QRectF QwtPlotSeriesItem::boundingRect() const [virtual]

Returns

An invalid bounding rect: QRectF(1.0, 1.0, -2.0, -2.0)

Note

A width or height < 0.0 is ignored by the autoscaler

Reimplemented from QwtPlotItem.

Reimplemented in QwtPlotTradingCurve, QwtPlotIntervalCurve, QwtPlotHistogram, QwtPlotBarChart, and QwtPlotMultiBarChart.

12.88.3.2 void QwtPlotSeriesItem::draw (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect) const [virtual]

Draw the complete series.

Parameters

| painter | Painter |
|------------|---------------------------------------|
| хМар | Maps x-values into pixel coordinates. |
| уМар | Maps y-values into pixel coordinates. |
| canvasRect | Contents rectangle of the canvas |

Implements QwtPlotItem.

12.88.3.3 virtual void QwtPlotSeriesItem::drawSeries (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [pure virtual]

Draw a subset of the samples

| painter | Painter |
|------------|---|
| хМар | Maps x-values into pixel coordinates. |
| уМар | Maps y-values into pixel coordinates. |
| canvasRect | Contents rectangle of the canvas |
| from | Index of the first point to be painted |
| to | Index of the last point to be painted. If to $<$ 0 the curve will be painted to its last point. |

Implemented in QwtPlotCurve, QwtPlotTradingCurve, QwtPlotIntervalCurve, QwtPlotHistogram, QwtPlotBarChart, QwtPlotMultiBarChart, and QwtPlotSpectroCurve.

12.88.3.4 Qt::Orientation QwtPlotSeriesItem::orientation () const

Returns

Orientation of the plot item

See Also

setOrientation()

12.88.3.5 void QwtPlotSeriesItem::setOrientation (Qt::Orientation orientation)

Set the orientation of the item.

The orientation() might be used in specific way by a plot item. F.e. a QwtPlotCurve uses it to identify how to display the curve int QwtPlotCurve::Steps or QwtPlotCurve::Sticks style.

See Also

orientation()

12.88.3.6 void QwtPlotSeriesItem::updateScaleDiv (const QwtScaleDiv & xScaleDiv, const QwtScaleDiv & yScaleDiv)
[virtual]

Update the item to changes of the axes scale division.

Update the item, when the axes of plot have changed. The default implementation does nothing, but items that depend on the scale division (like QwtPlotGrid()) have to reimplement updateScaleDiv()

updateScaleDiv() is only called when the ScaleInterest interest is enabled. The default implementation does nothing.

Parameters

| xScaleDiv | Scale division of the x-axis |
|-----------|------------------------------|
| yScaleDiv | Scale division of the y-axis |

See Also

QwtPlot::updateAxes(), ScaleInterest

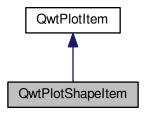
Reimplemented from QwtPlotItem.

12.89 QwtPlotShapeItem Class Reference

A plot item, which displays any graphical shape, that can be defined by a QPainterPath.

#include <qwt_plot_shapeitem.h>

Inheritance diagram for QwtPlotShapeItem:



Public Types

- enum PaintAttribute { ClipPolygons = 0x01 }
- enum LegendMode { LegendShape, LegendColor }

Mode how to display the item on the legend.

typedef QFlags< PaintAttribute > PaintAttributes

Paint attributes.

Public Member Functions

QwtPlotShapeItem (const QString &title=QString::null)

Constructor.

QwtPlotShapeItem (const QwtText &title)

Constructor.

virtual ~QwtPlotShapeItem ()

Destructor.

- void setPaintAttribute (PaintAttribute, bool on=true)
- · bool testPaintAttribute (PaintAttribute) const
- void setLegendMode (LegendMode)
- LegendMode legendMode () const
- void setRect (const QRectF &)

Set a path built from a rectangle.

• void setPolygon (const QPolygonF &)

Set a path built from a polygon.

void setShape (const QPainterPath &)

Set the shape to be displayed.

- QPainterPath shape () const
- void setPen (const QColor &, qreal width=0.0, Qt::PenStyle=Qt::SolidLine)
- void setPen (const QPen &)

Assign a pen.

- QPen pen () const
- void setBrush (const QBrush &)
- QBrush brush () const
- void setRenderTolerance (double)

Set the tolerance for the weeding optimization.

• double renderTolerance () const

- virtual QRectF boundingRect () const Bounding rectangle of the shape.
- virtual void draw (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &rect)
 const
- virtual QwtGraphic legendlcon (int index, const QSizeF &) const
- · virtual int rtti () const

Additional Inherited Members

12.89.1 Detailed Description

A plot item, which displays any graphical shape, that can be defined by a QPainterPath.

A QPainterPath is a shape composed from intersecting and uniting regions, rectangles, ellipses or irregular areas defined by lines, and curves. QwtPlotShapeItem displays a shape with a pen and brush.

QwtPlotShapeItem offers a couple of optimizations like clipping or weeding. These algorithms need to convert the painter path into polygons that might be less performant for paths built from curves and ellipses.

See Also

QwtPlotZone

12.89.2 Member Enumeration Documentation

12.89.2.1 enum QwtPlotShapeItem::LegendMode

Mode how to display the item on the legend.

Enumerator

LegendShape Display a scaled down version of the shape.

LegendColor Display a filled rectangle.

12.89.2.2 enum QwtPlotShapeItem::PaintAttribute

Attributes to modify the drawing algorithm. The default disables all attributes

See Also

setPaintAttribute(), testPaintAttribute()

Enumerator

ClipPolygons Clip polygons before painting them. In situations, where points are far outside the visible area (f.e when zooming deep) this might be a substantial improvement for the painting performance.
But polygon clipping will convert the painter path into polygons what might introduce a negative impact on the performance of paths composed from curves or ellipses.

12.89.3 Constructor & Destructor Documentation

12.89.3.1 QwtPlotShapeItem::QwtPlotShapeItem(const QString & title = QString::null) [explicit]

Constructor.

Sets the following item attributes:

• QwtPlotItem::AutoScale: true

QwtPlotItem::Legend: false

Parameters

| title | Title |
|-------|-------|
|-------|-------|

12.89.3.2 QwtPlotShapeItem::QwtPlotShapeItem (const QwtText & title) [explicit]

Constructor.

Sets the following item attributes:

• QwtPlotItem::AutoScale: true

• QwtPlotItem::Legend: false

Parameters

| title | Title |
|-------|-------|

12.89.4 Member Function Documentation

12.89.4.1 QBrush QwtPlotShapeItem::brush () const

Returns

Brush used to fill the shape

See Also

setBrush(), pen()

12.89.4.2 void QwtPlotShapeltem::draw (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect) const [virtual]

Draw the shape item

Parameters

| painter | Painter |
|------------|----------------------------------|
| хМар | X-Scale Map |
| уМар | Y-Scale Map |
| canvasRect | Contents rect of the plot canvas |

Implements QwtPlotItem.

12.89.4.3 QwtGraphic QwtPlotShapeltem::legendlcon (int index, const QSizeF & size) const [virtual]

Returns

A rectangle filled with the color of the brush (or the pen)

Parameters

| index | Index of the legend entry (usually there is only one) |
|-------|---|
| size | Icon size |

See Also

setLegendIconSize(), legendData()

Reimplemented from QwtPlotItem.

```
12.89.4.4 QwtPlotShapeItem::LegendMode QwtPlotShapeItem::legendMode ( ) const
Returns
     Mode how to represent the item on the legend
See Also
     legendMode()
12.89.4.5 QPen QwtPlotShapeItem::pen ( ) const
Returns
     Pen used to draw the outline of the shape
See Also
     setPen(), brush()
12.89.4.6 double QwtPlotShapeItem::renderTolerance ( ) const
Returns
     Tolerance for the weeding optimization
See Also
     setRenderTolerance()
12.89.4.7 int QwtPlotShapeItem::rtti() const [virtual]
Returns
     QwtPlotItem::Rtti_PlotShape
Reimplemented from QwtPlotItem.
12.89.4.8 void QwtPlotShapeItem::setBrush ( const QBrush & brush )
Assign a brush.
The brush is used to fill the path
Parameters
                     Brush
             brush
See Also
     brush(), pen()
12.89.4.9 void QwtPlotShapeItem::setLegendMode ( LegendMode mode )
Set the mode how to represent the item on the legend
```

Parameters

| mode | Mode |
|------|------|
|------|------|

See Also

legendMode()

12.89.4.10 void QwtPlotShapeItem::setPaintAttribute (PaintAttribute attribute, bool on = true)

Specify an attribute how to draw the shape

Parameters

| attribute | Paint attribute |
|-----------|-----------------|
| on | On/Off |

See Also

testPaintAttribute()

12.89.4.11 void QwtPlotShapeltem::setPen (const QColor & color, qreal width = 0.0, Qt::PenStyle style = Qt::SolidLine)

Build and assign a pen

In Qt5 the default pen width is 1.0 (0.0 in Qt4) what makes it non cosmetic (see QPen::isCosmetic()). This method has been introduced to hide this incompatibility.

Parameters

| color | Pen color |
|-------|-----------|
| width | Pen width |
| style | Pen style |

See Also

pen(), brush()

12.89.4.12 void QwtPlotShapeItem::setPen (const QPen & pen)

Assign a pen.

The pen is used to draw the outline of the shape

Parameters

| pen | Pen |
|-----|-----|

See Also

pen(), brush()

12.89.4.13 void QwtPlotShapeItem::setPolygon (const QPolygonF & polygon)

Set a path built from a polygon.

polygon Polygon

See Also

setShape(), setRect(), shape()

12.89.4.14 void QwtPlotShapeItem::setRect (const QRectF & rect)

Set a path built from a rectangle.

Parameters

rect Rectangle

See Also

setShape(), setPolygon(), shape()

12.89.4.15 void QwtPlotShapeItem::setRenderTolerance (double tolerance)

Set the tolerance for the weeding optimization.

After translating the shape into target device coordinate (usually widget geometries) the painter path can be simplified by a point weeding algorithm (Douglas-Peucker).

For shapes built from curves and ellipses weeding might have the opposite effect because they have to be expanded to polygons.

Parameters

tolerance Accepted error when reducing the number of points A value \leq 0.0 disables weeding.

See Also

renderTolerance(), QwtWeedingCurveFitter

12.89.4.16 void QwtPlotShapeItem::setShape (const QPainterPath & shape)

Set the shape to be displayed.

Parameters

shape | Shape

See Also

setShape(), shape()

Returns

Shape to be displayed

See Also

setShape()

12.89.4.18 bool QwtPlotShapeItem::testPaintAttribute (PaintAttribute attribute) const

Returns

True, when attribute is enabled

See Also

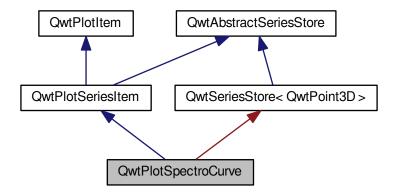
setPaintAttribute()

12.90 QwtPlotSpectroCurve Class Reference

Curve that displays 3D points as dots, where the z coordinate is mapped to a color.

#include <qwt_plot_spectrocurve.h>

Inheritance diagram for QwtPlotSpectroCurve:



Public Types

• enum PaintAttribute { ClipPoints = 1 }

Paint attributes.

typedef QFlags< PaintAttribute > PaintAttributes

Paint attributes.

Public Member Functions

- QwtPlotSpectroCurve (const QString &title=QString::null)
- QwtPlotSpectroCurve (const QwtText &title)
- virtual ~QwtPlotSpectroCurve ()

Destructor.

- · virtual int rtti () const
- void setPaintAttribute (PaintAttribute, bool on=true)
- · bool testPaintAttribute (PaintAttribute) const
- void setSamples (const QVector< QwtPoint3D > &)
- void setSamples (QwtSeriesData < QwtPoint3D > *)
- void setColorMap (QwtColorMap *)

- const QwtColorMap * colorMap () const
- void setColorRange (const QwtInterval &)
- · QwtInterval & colorRange () const
- virtual void drawSeries (QPainter *, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int from, int to) const
- void setPenWidth (double width)
- double penWidth () const

Protected Member Functions

 virtual void drawDots (QPainter *, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int from, int to) const

12.90.1 Detailed Description

Curve that displays 3D points as dots, where the z coordinate is mapped to a color.

12.90.2 Member Enumeration Documentation

12.90.2.1 enum QwtPlotSpectroCurve::PaintAttribute

Paint attributes.

Enumerator

ClipPoints Clip points outside the canvas rectangle.

12.90.3 Constructor & Destructor Documentation

12.90.3.1 QwtPlotSpectroCurve::QwtPlotSpectroCurve(const QString & title = QString::null) [explicit]

Constructor

Parameters

| title | Title of the curve |
|-------|--------------------|

12.90.3.2 QwtPlotSpectroCurve::QwtPlotSpectroCurve (const QwtText & title) [explicit]

Constructor

Parameters

```
title Title of the curve
```

12.90.4 Member Function Documentation

12.90.4.1 const QwtColorMap * QwtPlotSpectroCurve::colorMap () const

Returns

Color Map used for mapping the intensity values to colors

See Also

setColorMap(), setColorRange(), QwtColorMap::color()

12.90.4.2 QwtInterval & QwtPlotSpectroCurve::colorRange () const

Returns

Value interval, that corresponds to the color map

See Also

setColorRange(), setColorMap(), QwtColorMap::color()

12.90.4.3 void QwtPlotSpectroCurve::drawDots (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [protected], [virtual]

Draw a subset of the points

Parameters

| painter | Painter | |
|--|--|--|
| xMap Maps x-values into pixel coordinates. | | |
| уМар | Maps y-values into pixel coordinates. | |
| canvasRect | Contents rectangle of the canvas | |
| from | Index of the first sample to be painted | |
| to | Index of the last sample to be painted. If to $<$ 0 the series will be painted to its last sample. | |

See Also

drawSeries()

12.90.4.4 void QwtPlotSpectroCurve::drawSeries (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [virtual]

Draw a subset of the points

Parameters

| painter | Painter | |
|---|--|--|
| хМар | Maps x-values into pixel coordinates. | |
| уМар | Maps y-values into pixel coordinates. | |
| canvasRect Contents rectangle of the canvas | | |
| from | Index of the first sample to be painted | |
| to | Index of the last sample to be painted. If to $<$ 0 the series will be painted to its last sample. | |

See Also

drawDots()

Implements QwtPlotSeriesItem.

12.90.4.5 double QwtPlotSpectroCurve::penWidth () const

Returns

Pen width used to draw a dot

See Also

setPenWidth()

12.90.4.6 int QwtPlotSpectroCurve::rtti() const [virtual]

Returns

QwtPlotItem::Rtti_PlotSpectroCurve

Reimplemented from QwtPlotItem.

12.90.4.7 void QwtPlotSpectroCurve::setColorMap (QwtColorMap * colorMap)

Change the color map

Often it is useful to display the mapping between intensities and colors as an additional plot axis, showing a color bar.

Parameters

| colorMap | Color Map | | |
|----------|-----------|--|--|

See Also

colorMap(), setColorRange(), QwtColorMap::color(), QwtScaleWidget::setColorBarEnabled(), QwtScaleWidget::setColorMap()

12.90.4.8 void QwtPlotSpectroCurve::setColorRange (const QwtInterval & interval)

Set the value interval, that corresponds to the color map

Parameters

| interval interval.minValue() corresponds to 0.0, interval.maxValue() to 1.0 on the color map. | |
|---|--|
|---|--|

See Also

colorRange(), setColorMap(), QwtColorMap::color()

12.90.4.9 void QwtPlotSpectroCurve::setPaintAttribute (PaintAttribute attribute, bool on = true)

Specify an attribute how to draw the curve

Parameters

| attribute | Paint attribute |
|-----------|---|
| on | On/Off /sa PaintAttribute, testPaintAttribute() |

12.90.4.10 void QwtPlotSpectroCurve::setPenWidth (double penWidth)

Assign a pen width

Parameters

| penWidth | New pen width |
|----------|---------------|

See Also

penWidth()

12.90.4.11 void QwtPlotSpectroCurve::setSamples (const QVector < QwtPoint3D > & samples)

Initialize data with an array of samples.

Parameters

| samples | Vector of points |
|---------|------------------|

12.90.4.12 void QwtPlotSpectroCurve::setSamples (QwtSeriesData < QwtPoint3D > * data)

Assign a series of samples

setSamples() is just a wrapper for setData() without any additional value - beside that it is easier to find for the developer.

Parameters

| data | Data |
|------|------|

Warning

The item takes ownership of the data object, deleting it when its not used anymore.

12.90.4.13 bool QwtPlotSpectroCurve::testPaintAttribute (PaintAttribute attribute) const

Returns

True, when attribute is enabled

See Also

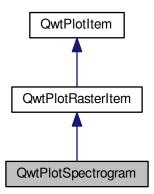
PaintAttribute, setPaintAttribute()

12.91 QwtPlotSpectrogram Class Reference

A plot item, which displays a spectrogram.

#include <qwt_plot_spectrogram.h>

Inheritance diagram for QwtPlotSpectrogram:



Public Types

enum DisplayMode { ImageMode = 0x01, ContourMode = 0x02 }

typedef QFlags < DisplayMode > DisplayModes

Display modes.

Public Member Functions

- QwtPlotSpectrogram (const QString &title=QString::null)
- virtual ~QwtPlotSpectrogram ()

Destructor.

- void setDisplayMode (DisplayMode, bool on=true)
- bool testDisplayMode (DisplayMode) const
- void setData (QwtRasterData *data)
- const QwtRasterData * data () const
- QwtRasterData * data ()
- void setColorMap (QwtColorMap *)
- const QwtColorMap * colorMap () const
- virtual QwtInterval interval (Qt::Axis) const
- virtual QRectF pixelHint (const QRectF &) const

Pixel hint.

- void setDefaultContourPen (const QColor &, greal width=0.0, Qt::PenStyle=Qt::SolidLine)
- void setDefaultContourPen (const QPen &)

Set the default pen for the contour lines.

- QPen defaultContourPen () const
- · virtual QPen contourPen (double level) const

Calculate the pen for a contour line.

- void setConrecFlag (QwtRasterData::ConrecFlag, bool on)
- · bool testConrecFlag (QwtRasterData::ConrecFlag) const
- void setContourLevels (const QList< double > &)
- QList< double > contourLevels () const
- · virtual int rtti () const
- virtual void draw (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &rect)
 const

Draw the spectrogram.

Protected Member Functions

 virtual Qlmage renderlmage (const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &area, const QSize &imageSize) const

Render an image from data and color map.

• virtual QSize contourRasterSize (const QRectF &, const QRect &) const

Return the raster to be used by the CONREC contour algorithm.

- · virtual QwtRasterData::ContourLines renderContourLines (const QRectF &rect, const QSize &raster) const
- virtual void drawContourLines (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QwtRasterData::ContourLines &lines) const
- void renderTile (const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRect &imageRect, QImage *image) const

Render a tile of an image.

12.91.1 Detailed Description

A plot item, which displays a spectrogram.

A spectrogram displays 3-dimensional data, where the 3rd dimension (the intensity) is displayed using colors. The colors are calculated from the values using a color map.

On multi-core systems the performance of the image composition can often be improved by dividing the area into tiles - each of them rendered in a different thread (see QwtPlotItem::setRenderThreadCount()).

In ContourMode contour lines are painted for the contour levels.

See Also

QwtRasterData, QwtColorMap, QwtPlotItem::setRenderThreadCount()

12.91.2 Member Enumeration Documentation

12.91.2.1 enum QwtPlotSpectrogram::DisplayMode

The display mode controls how the raster data will be represented.

See Also

setDisplayMode(), testDisplayMode()

Enumerator

ImageMode The values are mapped to colors using a color map.

ContourMode The data is displayed using contour lines.

12.91.3 Constructor & Destructor Documentation

12.91.3.1 QwtPlotSpectrogram::QwtPlotSpectrogram (const QString & title = QString::null) [explicit]

Sets the following item attributes:

- QwtPlotItem::AutoScale: true
- QwtPlotItem::Legend: false

The z value is initialized by 8.0.

Parameters

| title | Title |
|-------|-------|

See Also

QwtPlotItem::setItemAttribute(), QwtPlotItem::setZ()

12.91.4 Member Function Documentation

12.91.4.1 const QwtColorMap * QwtPlotSpectrogram::colorMap () const

Returns

Color Map used for mapping the intensity values to colors

See Also

setColorMap()

12.91.4.2 QList< double > QwtPlotSpectrogram::contourLevels () const

Returns

Levels of the contour lines.

The levels are sorted in increasing order.

See Also

contourLevels(), renderContourLines(), QwtRasterData::contourLines()

12.91.4.3 QPen QwtPlotSpectrogram::contourPen (double level) const [virtual]

Calculate the pen for a contour line.

The color of the pen is the color for level calculated by the color map

Parameters

| level | Contour level |
|-------|---------------|

Returns

Pen for the contour line

Note

contourPen is only used if defaultContourPen().style() == Qt::NoPen

See Also

setDefaultContourPen(), setColorMap(), setContourLevels()

12.91.4.4 QSize QwtPlotSpectrogram::contourRasterSize (const QRectF & area, const QRect & rect) const [protected], [virtual]

Return the raster to be used by the CONREC contour algorithm.

A larger size will improve the precision of the CONREC algorithm, but will slow down the time that is needed to calculate the lines.

The default implementation returns rect.size() / 2 bounded to the resolution depending on pixelSize().

Parameters

| area | Rectangle, where to calculate the contour lines | |
|------|--|--|
| rect | Rectangle in pixel coordinates, where to paint the contour lines | |

Returns

Raster to be used by the CONREC contour algorithm.

Note

The size will be bounded to rect.size().

See Also

drawContourLines(), QwtRasterData::contourLines()

```
12.91.4.5 const QwtRasterData * QwtPlotSpectrogram::data ( ) const

Returns

Spectrogram data

See Also
setData()

12.91.4.6 QwtRasterData * QwtPlotSpectrogram::data ( )

Returns
Spectrogram data

See Also
setData()
```

12.91.4.7 QPen QwtPlotSpectrogram::defaultContourPen () const

Returns

Default contour pen

See Also

setDefaultContourPen()

12.91.4.8 void QwtPlotSpectrogram::draw (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect) const [virtual]

Draw the spectrogram.

Parameters

| painter | Painter |
|------------|---|
| хМар | Maps x-values into pixel coordinates. |
| уМар | Maps y-values into pixel coordinates. |
| canvasRect | Contents rectangle of the canvas in painter coordinates |

See Also

setDisplayMode(), renderImage(), QwtPlotRasterItem::draw(), drawContourLines()

Reimplemented from QwtPlotRasterItem.

12.91.4.9 void QwtPlotSpectrogram::drawContourLines (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QwtRasterData::ContourLines & contourLines) const [protected], [virtual]

Paint the contour lines

| painter | Painter |
|--------------|---------------------------------------|
| хМар | Maps x-values into pixel coordinates. |
| уМар | Maps y-values into pixel coordinates. |
| contourLines | Contour lines |

See Also

renderContourLines(), defaultContourPen(), contourPen()

12.91.4.10 QwtInterval QwtPlotSpectrogram::interval (Qt::Axis axis) const [virtual]

Returns

Bounding interval for an axis

The default implementation returns the interval of the associated raster data object.

Parameters

| axis | X, Y, or Z axis |
|------|-----------------|

See Also

QwtRasterData::interval()

Reimplemented from QwtPlotRasterItem.

12.91.4.11 QRectF QwtPlotSpectrogram::pixelHint (const QRectF & area) const [virtual]

Pixel hint.

The geometry of a pixel is used to calculated the resolution and alignment of the rendered image.

The default implementation returns data()->pixelHint(rect);

Parameters

| n most implementations the resolution of the data doesn' | depend on the requested area. |
|--|-------------------------------|
|--|-------------------------------|

Returns

Bounding rectangle of a pixel

See Also

QwtPlotRasterItem::pixelHint(), QwtRasterData::pixelHint(), render(), renderImage()

Reimplemented from QwtPlotRasterItem.

12.91.4.12 QwtRasterData::ContourLines QwtPlotSpectrogram::renderContourLines (const QRectF & rect, const QSize & raster) const [protected], [virtual]

Calculate contour lines

Parameters

| rect | Rectangle, where to calculate the contour lines |
|--------|---|
| raster | Raster, used by the CONREC algorithm |

Returns

Calculated contour lines

See Also

contourLevels(), setConrecFlag(), QwtRasterData::contourLines()

12.91.4.13 QImage QwtPlotSpectrogram::renderImage (const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & area, const QSize & imageSize) const [protected], [virtual]

Render an image from data and color map.

For each pixel of area the value is mapped into a color.

Parameters

| хМар | X-Scale Map |
|-----------|---|
| уМар | Y-Scale Map |
| area | Requested area for the image in scale coordinates |
| imageSize | Size of the requested image |

Returns

A QImage::Format Indexed8 or QImage::Format ARGB32 depending on the color map.

See Also

QwtRasterData::value(), QwtColorMap::rgb(), QwtColorMap::colorIndex()

Implements QwtPlotRasterItem.

12.91.4.14 void QwtPlotSpectrogram::renderTile (const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRect & tile, Qlmage * image) const [protected]

Render a tile of an image.

Rendering in tiles can be used to composite an image in parallel threads.

Parameters

| хМар | X-Scale Map |
|-------|---|
| уМар | Y-Scale Map |
| tile | Geometry of the tile in image coordinates |
| image | Image to be rendered |

12.91.4.15 int QwtPlotSpectrogram::rtti()const [virtual]

Returns

QwtPlotItem::Rtti_PlotSpectrogram

Reimplemented from QwtPlotItem.

12.91.4.16 void QwtPlotSpectrogram::setColorMap (QwtColorMap * colorMap)

Change the color map

Often it is useful to display the mapping between intensities and colors as an additional plot axis, showing a color bar.

| colorMap | Color Map |
|----------|-----------|
|----------|-----------|

See Also

 $color Map(), \ Qwt Scale Widget:: set Color Bar Enabled(), \ Qwt Scale Widget:: set Color Map()$

12.91.4.17 void QwtPlotSpectrogram::setConrecFlag (QwtRasterData::ConrecFlag flag, bool on)

Modify an attribute of the CONREC algorithm, used to calculate the contour lines.

Parameters

| flag | CONREC flag |
|------|-------------|
| on | On/Off |

See Also

testConrecFlag(), renderContourLines(), QwtRasterData::contourLines()

12.91.4.18 void QwtPlotSpectrogram::setContourLevels (const QList< double > & levels)

Set the levels of the contour lines

Parameters

| levels | Values of the contour levels |
|--------|------------------------------|

See Also

contourLevels(), renderContourLines(), QwtRasterData::contourLines()

Note

contourLevels returns the same levels but sorted.

12.91.4.19 void QwtPlotSpectrogram::setData (QwtRasterData * data)

Set the data to be displayed

Parameters

| data | Spectrogram Data |
|------|------------------|

See Also

data()

12.91.4.20 void QwtPlotSpectrogram::setDefaultContourPen (const QColor & color, qreal width = 0.0, Qt::PenStyle style = Qt::SolidLine)

Build and assign the default pen for the contour lines

In Qt5 the default pen width is 1.0 (0.0 in Qt4) what makes it non cosmetic (see QPen::isCosmetic()). This method has been introduced to hide this incompatibility.

Parameters

| color | Pen color |
|-------|-----------|
| width | Pen width |
| style | Pen style |

See Also

pen(), brush()

12.91.4.21 void QwtPlotSpectrogram::setDefaultContourPen (const QPen & pen)

Set the default pen for the contour lines.

If the spectrogram has a valid default contour pen a contour line is painted using the default contour pen. Otherwise (pen.style() == Qt::NoPen) the pen is calculated for each contour level using contourPen().

See Also

defaultContourPen(), contourPen()

12.91.4.22 void QwtPlotSpectrogram::setDisplayMode (DisplayMode mode, bool on = true)

The display mode controls how the raster data will be represented.

Parameters

| mode | Display mode |
|------|--------------|
| on | On/Off |

The default setting enables ImageMode.

See Also

DisplayMode, displayMode()

12.91.4.23 bool QwtPlotSpectrogram::testConrecFlag (QwtRasterData::ConrecFlag flag) const

Test an attribute of the CONREC algorithm, used to calculate the contour lines.

Parameters

| flag CONI | IREC flag |
|-----------|-----------|
|-----------|-----------|

Returns

true, is enabled

The default setting enables QwtRasterData::IgnoreAllVerticesOnLevel

See Also

setConrecClag(), renderContourLines(), QwtRasterData::contourLines()

12.91.4.24 bool QwtPlotSpectrogram::testDisplayMode (DisplayMode mode) const

The display mode controls how the raster data will be represented.

| mode | Display mode |
|------|--------------|
|------|--------------|

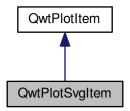
Returns

true if mode is enabled

12.92 QwtPlotSvgItem Class Reference

A plot item, which displays data in Scalable Vector Graphics (SVG) format.

Inheritance diagram for QwtPlotSvgItem:



Public Member Functions

• QwtPlotSvgItem (const QString &title=QString::null)

Constructor.

QwtPlotSvgItem (const QwtText &title)

Constructor.

• virtual \sim QwtPlotSvgItem ()

Destructor.

- bool loadFile (const QRectF &, const QString &fileName)
- bool loadData (const QRectF &, const QByteArray &)
- virtual QRectF boundingRect () const

Bounding rectangle of the item.

- virtual void draw (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &rect)
 const
- · virtual int rtti () const

Protected Member Functions

- · const QSvgRenderer & renderer () const
- QSvgRenderer & renderer ()
- void render (QPainter *painter, const QRectF &viewBox, const QRectF &rect) const
- QRectF viewBox (const QRectF & area) const

Additional Inherited Members

12.92.1 Detailed Description

A plot item, which displays data in Scalable Vector Graphics (SVG) format.

SVG images are often used to display maps

12.92.2 Constructor & Destructor Documentation

12.92.2.1 QwtPlotSvgltem::QwtPlotSvgltem (const QString & title = QString::null) [explicit]

Constructor.

Sets the following item attributes:

• QwtPlotItem::AutoScale: true

• QwtPlotItem::Legend: false

Parameters

| - | | |
|---|-------|-------|
| | title | Title |

12.92.2.2 QwtPlotSvgltem::QwtPlotSvgltem (const QwtText & title) [explicit]

Constructor.

Sets the following item attributes:

· QwtPlotItem::AutoScale: true

• QwtPlotItem::Legend: false

Parameters

| title | Title |
|-------|-------|

12.92.3 Member Function Documentation

12.92.3.1 void QwtPlotSvgltem::draw (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect) const [virtual]

Draw the SVG item

Parameters

| | painter | Painter |
|---|------------|----------------------------------|
| | хМар | X-Scale Map |
| | уМар | Y-Scale Map |
| Ī | canvasRect | Contents rect of the plot canvas |

Implements QwtPlotItem.

12.92.3.2 bool QwtPlotSvgltem::loadData (const QRectF & rect, const QByteArray & data)

Load SVG data

| rect | Bounding rectangle |
|------|--------------------|
| data | in SVG format |

Returns

true, if the SVG data could be loaded

12.92.3.3 bool QwtPlotSvgItem::loadFile (const QRectF & rect, const QString & fileName)

Load a SVG file

Parameters

| rect | Bounding rectangle |
|----------|--------------------|
| fileName | SVG file name |

Returns

true, if the SVG file could be loaded

12.92.3.4 void QwtPlotSvgltem::render (QPainter * painter, const QRectF & viewBox, const QRectF & rect) const [protected]

Render the SVG data

Parameters

| painter | Painter |
|---------|---------------------------------------|
| viewBox | View Box, see QSvgRenderer::viewBox() |
| rect | Target rectangle on the paint device |

12.92.3.5 const QSvgRenderer & QwtPlotSvgltem::renderer () const [protected]

Returns

Renderer used to render the SVG data

12.92.3.6 QSvgRenderer & QwtPlotSvgItem::renderer () [protected]

Returns

Renderer used to render the SVG data

12.92.3.7 int QwtPlotSvgltem::rtti() const [virtual]

Returns

QwtPlotItem::Rtti_PlotSVG

Reimplemented from QwtPlotItem.

12.92.3.8 QRectF QwtPlotSvgltem::viewBox (const QRectF & rect) const [protected]

Calculate the view box from rect and boundingRect().

Parameters

| rect | Rectangle in scale coordinates | |
|------|--------------------------------|--|
|------|--------------------------------|--|

Returns

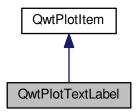
View box, see QSvgRenderer::viewBox()

12.93 QwtPlotTextLabel Class Reference

A plot item, which displays a text label.

```
#include <qwt_plot_textlabel.h>
```

Inheritance diagram for QwtPlotTextLabel:



Public Member Functions

• QwtPlotTextLabel ()

Constructor.

virtual ~QwtPlotTextLabel ()

Destructor.

- virtual int rtti () const
- void setText (const QwtText &)
- QwtText text () const
- void setMargin (int margin)
- int margin () const
- virtual QRectF textRect (const QRectF &, const QSizeF &) const

Align the text label.

Protected Member Functions

- virtual void draw (QPainter *, const QwtScaleMap &, const QwtScaleMap &, const QRectF &) const
- void invalidateCache ()

Invalidate all internal cache.

Additional Inherited Members

12.93.1 Detailed Description

A plot item, which displays a text label.

QwtPlotTextLabel displays a text label aligned to the plot canvas.

In opposite to QwtPlotMarker the position of the label is unrelated to plot coordinates.

As drawing a text is an expensive operation the label is cached in a pixmap to speed up replots.

Example

The following code shows how to add a title.

```
QwtText title( "Plot Title" );
title.setRenderFlags( Qt::AlignHCenter | Qt::AlignTop );

QFont font;
font.setBold( true );
title.setFont( font );

QwtPlotTextLabel *titleItem = new QwtPlotTextLabel();
titleItem->setText( title );
titleItem->attach( this );
```

See Also

QwtPlotMarker

12.93.2 Constructor & Destructor Documentation

```
12.93.2.1 QwtPlotTextLabel::QwtPlotTextLabel()
```

Constructor.

Initializes an text label with an empty text

Sets the following item attributes:

QwtPlotItem::AutoScale: true

• QwtPlotItem::Legend: false

The z value is initialized by 150

See Also

QwtPlotItem::setItemAttribute(), QwtPlotItem::setZ()

12.93.3 Member Function Documentation

12.93.3.1 void QwtPlotTextLabel::draw (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect) const [protected], [virtual]

Draw the text label

Parameters

| painter | Painter |
|---------|-------------|
| хМар | x Scale Map |
| уМар | y Scale Map |

Contents rectangle of the canvas in painter coordinates canvasRect See Also textRect() Implements QwtPlotItem. 12.93.3.2 int QwtPlotTextLabel::margin () const Returns Margin added to the contentsMargins() of the canvas See Also setMargin() 12.93.3.3 int QwtPlotTextLabel::rtti() const [virtual] Returns QwtPlotItem::Rtti_PlotTextLabel Reimplemented from QwtPlotItem. 12.93.3.4 void QwtPlotTextLabel::setMargin (int margin) Set the margin The margin is the distance between the contentsRect() of the plot canvas and the rectangle where the label can be displayed. **Parameters** margin Margin See Also margin(), textRect() 12.93.3.5 void QwtPlotTextLabel::setText (const QwtText & text) Set the text The label will be aligned to the plot canvas according to the alignment flags of text. **Parameters** Text to be displayed text See Also text(), QwtText::renderFlags() 12.93.3.6 QwtText QwtPlotTextLabel::text () const Returns Text to be displayed See Also setText()

12.93.3.7 QRectF QwtPlotTextLabel::textRect (const QRectF & rect, const QSizeF & textSize) const [virtual] Align the text label.

Parameters

| rect | Canvas rectangle with margins subtracted |
|----------|--|
| textSize | Size required to draw the text |

Returns

A rectangle aligned according the the alignment flags of the text.

See Also

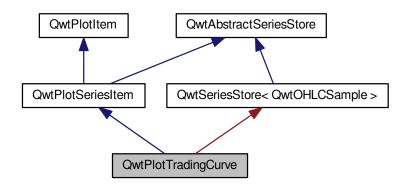
setMargin(), QwtText::renderFlags(), QwtText::textSize()

12.94 QwtPlotTradingCurve Class Reference

QwtPlotTradingCurve illustrates movements in the price of a financial instrument over time.

#include <qwt_plot_tradingcurve.h>

Inheritance diagram for QwtPlotTradingCurve:



Public Types

- enum SymbolStyle { NoSymbol = -1, Bar, CandleStick, UserSymbol = 100 }
 Symbol styles.
- enum Direction { Increasing, Decreasing }

Direction of a price movement.

- enum PaintAttribute { ClipSymbols = 0x01 }
- typedef QFlags < PaintAttribute > PaintAttributes

Paint attributes.

Public Member Functions

- QwtPlotTradingCurve (const QString &title=QString::null)
- QwtPlotTradingCurve (const QwtText &title)
- virtual ~QwtPlotTradingCurve ()

Destructor.

- virtual int rtti () const
- void setPaintAttribute (PaintAttribute, bool on=true)
- · bool testPaintAttribute (PaintAttribute) const
- void setSamples (const QVector< QwtOHLCSample > &)
- void setSamples (QwtSeriesData < QwtOHLCSample > *)
- void setSymbolStyle (SymbolStyle style)
- SymbolStyle symbolStyle () const
- void setSymbolPen (const QColor &, greal width=0.0, Qt::PenStyle=Qt::SolidLine)
- void setSymbolPen (const QPen &)

Set the symbol pen.

- QPen symbolPen () const
- void setSymbolBrush (Direction, const QBrush &)
- QBrush symbolBrush (Direction) const
- void setSymbolExtent (double width)

Set the extent of the symbol.

- · double symbolExtent () const
- void setMinSymbolWidth (double)
- double minSymbolWidth () const
- void setMaxSymbolWidth (double)
- double maxSymbolWidth () const
- virtual void drawSeries (QPainter *painter, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect, int from, int to) const
- virtual QRectF boundingRect () const
- virtual QwtGraphic legendlcon (int index, const QSizeF &) const

Protected Member Functions

void init ()

Initialize internal members.

- virtual void drawSymbols (QPainter *, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const Q-RectF &canvasRect, int from, int to) const
- virtual void drawUserSymbol (QPainter *, SymbolStyle, const QwtOHLCSample &, Qt::Orientation, bool inverted, double width) const

Draw a symbol for a symbol style >= UserSymbol.

 void drawBar (QPainter *painter, const QwtOHLCSample &, Qt::Orientation, bool inverted, double width) const

Draw a bar.

- void drawCandleStick (QPainter *, const QwtOHLCSample &, Qt::Orientation, double width) const
 Draw a candle stick.
- virtual double scaledSymbolWidth (const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRectF &canvasRect) const

12.94.1 Detailed Description

QwtPlotTradingCurve illustrates movements in the price of a financial instrument over time.

QwtPlotTradingCurve supports candlestick or bar (OHLC) charts that are used in the domain of technical analysis.

While the length (height or width depending on orientation()) of each symbol depends on the corresponding OHLC sample the size of the other dimension can be controlled using:

- setSymbolExtent()
- setSymbolMinWidth()

setSymbolMaxWidth()

The extent is a size in scale coordinates, so that the symbol width is increasing when the plot is zoomed in. Minimum/Maximum width is in widget coordinates independent from the zoom level. When setting the minimum and maximum to the same value, the width of the symbol is fixed.

12.94.2 Member Enumeration Documentation

12.94.2.1 enum QwtPlotTradingCurve::Direction

Direction of a price movement.

Enumerator

Increasing The closing price is higher than the opening price.

Decreasing The closing price is lower than the opening price.

12.94.2.2 enum QwtPlotTradingCurve::PaintAttribute

Attributes to modify the drawing algorithm.

See Also

setPaintAttribute(), testPaintAttribute()

Enumerator

ClipSymbols Check if a symbol is on the plot canvas before painting it.

12.94.2.3 enum QwtPlotTradingCurve::SymbolStyle

Symbol styles.

The default setting is QwtPlotSeriesItem::CandleStick.

See Also

setSymbolStyle(), symbolStyle()

Enumerator

NoSymbol Nothing is displayed.

Bar A line on the chart shows the price range (the highest and lowest prices) over one unit of time, e.g. one day or one hour. Tick marks project from each side of the line indicating the opening and closing price.

CandleStick The range between opening/closing price are displayed as a filled box. The fill brush depends on the direction of the price movement. The box is connected to the highest/lowest values by lines.

UserSymbol SymbolTypes >= UserSymbol are displayed by drawUserSymbol(), that needs to be overloaded and implemented in derived curve classes.

See Also

drawUserSymbol()

12.94.3 Constructor & Destructor Documentation

12.94.3.1 QwtPlotTradingCurve::QwtPlotTradingCurve (const QString & title = QString::null) [explicit]

Constructor

| title | Title of the curve |
|-------|--------------------|
|-------|--------------------|

12.94.3.2 QwtPlotTradingCurve::QwtPlotTradingCurve (const QwtText & title) [explicit]

Constructor

Parameters

| title | Title of the curve |
|-------|--------------------|

12.94.4 Member Function Documentation

12.94.4.1 QRectF QwtPlotTradingCurve::boundingRect() const [virtual]

Returns

Bounding rectangle of all samples. For an empty series the rectangle is invalid.

Reimplemented from QwtPlotSeriesItem.

12.94.4.2 void QwtPlotTradingCurve::drawBar (QPainter * painter, const QwtOHLCSample & sample, Qt::Orientation orientation, bool inverted, double width) const [protected]

Draw a bar.

Parameters

| painter | Qt painter, initialized with pen/brush |
|-------------|---|
| sample | Sample, already translated into paint device coordinates |
| orientation | Vertical or horizontal |
| inverted | When inverted is false the open tick is painted to the left/top, otherwise it is painted |
| | right/bottom. The close tick is painted in the opposite direction of the open tick. painted |
| | in the opposite d opposite direction. |
| width | Width or height of the candle, depending on the orientation |

See Also

Bar

12.94.4.3 void QwtPlotTradingCurve::drawCandleStick (QPainter * painter, const QwtOHLCSample & sample, Qt::Orientation orientation, double width) const [protected]

Draw a candle stick.

Parameters

| painter | Qt painter, initialized with pen/brush |
|-------------|---|
| sample | Samples already translated into paint device coordinates |
| orientation | Vertical or horizontal |
| width | Width or height of the candle, depending on the orientation |

See Also

CandleStick

12.94.4.4 void QwtPlotTradingCurve::drawSeries (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [virtual]

Draw an interval of the curve

Parameters

| painter | Painter |
|------------|---|
| хМар | Maps x-values into pixel coordinates. |
| уМар | Maps y-values into pixel coordinates. |
| canvasRect | Contents rectangle of the canvas |
| from | Index of the first point to be painted |
| to | Index of the last point to be painted. If to $<$ 0 the curve will be painted to its last point. |

See Also

drawSymbols()

Implements QwtPlotSeriesItem.

12.94.4.5 void QwtPlotTradingCurve::drawSymbols (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect, int from, int to) const [protected], [virtual]

Draw symbols

Parameters

| painter | Painter |
|------------|--|
| хМар | x map |
| уМар | y map |
| canvasRect | Contents rectangle of the canvas |
| from | Index of the first point to be painted |
| to | Index of the last point to be painted |

See Also

drawSeries()

12.94.4.6 void QwtPlotTradingCurve::drawUserSymbol (QPainter * painter, SymbolStyle symbolStyle, const QwtOHLCSample & sample, Qt::Orientation orientation, bool inverted, double symbolWidth) const [protected], [virtual]

Draw a symbol for a symbol style >= UserSymbol.

The implementation does nothing and is intended to be overloaded

Parameters

| painter | Qt painter, initialized with pen/brush |
|-------------|--|
| symbolStyle | Symbol style |
| sample | Samples already translated into paint device coordinates |
| orientation | Vertical or horizontal |
| inverted | True, when the opposite scale (Qt::Vertical: x, Qt::Horizontal: y) is increasing in the opposite |
| | direction as QPainter coordinates. |
| symbolWidth | Width of the symbol in paint device coordinates |

12.94.4.7 QwtGraphic QwtPlotTradingCurve::legendlcon(int index, const QSizeF & size) const [virtual]

Returns

A rectangle filled with the color of the symbol pen

| index | Index of the legend entry (usually there is only one) |
|-------|---|
| size | Icon size |

See Also

setLegendIconSize(), legendData()

Reimplemented from QwtPlotItem.

12.94.4.8 double QwtPlotTradingCurve::maxSymbolWidth () const

Returns

Maximum for the symbol width

See Also

setMaxSymbolWidth(), minSymbolWidth(), symbolExtent()

12.94.4.9 double QwtPlotTradingCurve::minSymbolWidth () const

Returns

Minmum for the symbol width

See Also

setMinSymbolWidth(), maxSymbolWidth(), symbolExtent()

12.94.4.10 int QwtPlotTradingCurve::rtti() const [virtual]

Returns

QwtPlotItem::Rtti_PlotTradingCurve

Reimplemented from QwtPlotItem.

12.94.4.11 double QwtPlotTradingCurve::scaledSymbolWidth (const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect) const [protected], [virtual]

Calculate the symbol width in paint coordinates

The width is calculated by scaling the symbol extent into paint device coordinates bounded by the minimum/maximum symbol width.

Parameters

| хМар | Maps x-values into pixel coordinates. |
|------------|---------------------------------------|
| уМар | Maps y-values into pixel coordinates. |
| canvasRect | Contents rectangle of the canvas |

Returns

Symbol width in paint coordinates

See Also

symbolExtent(), minSymbolWidth(), maxSymbolWidth()

12.94.4.12 void QwtPlotTradingCurve::setMaxSymbolWidth (double $\it width$)

Set a maximum for the symbol width

A value <= 0.0 means an unlimited width

| width | Width in paint device coordinates |
|-------|-----------------------------------|
|-------|-----------------------------------|

See Also

maxSymbolWidth(), setMinSymbolWidth(), setSymbolExtent()

12.94.4.13 void QwtPlotTradingCurve::setMinSymbolWidth (double width)

Set a minimum for the symbol width

Parameters

| width | Width in paint device coordinates |
|-------|-----------------------------------|

See Also

minSymbolWidth(), setMaxSymbolWidth(), setSymbolExtent()

12.94.4.14 void QwtPlotTradingCurve::setPaintAttribute (PaintAttribute attribute, bool on = true)

Specify an attribute how to draw the curve

Parameters

| attribute | Paint attribute |
|-----------|-----------------|
| on | On/Off |

See Also

testPaintAttribute()

12.94.4.15 void QwtPlotTradingCurve::setSamples (const QVector < QwtOHLCSample > & samples)

Initialize data with an array of samples.

Parameters

| samples | Vector of samples |
|---------|-------------------|

See Also

QwtPlotSeriesItem::setData()

12.94.4.16 void QwtPlotTradingCurve::setSamples (QwtSeriesData < QwtOHLCSample > * data)

Assign a series of samples

setSamples() is just a wrapper for setData() without any additional value - beside that it is easier to find for the developer.

Parameters

| data | Data |
|------|------|

Warning

The item takes ownership of the data object, deleting it when its not used anymore.

12.94.4.17 void QwtPlotTradingCurve::setSymbolBrush (Direction direction, const QBrush & brush)

Set the symbol brush

Parameters

| direction | Direction type |
|-----------|---|
| brush | Brush used to fill the body of all candlestick symbols with the direction |

See Also

symbolBrush(), setSymbolPen()

12.94.4.18 void QwtPlotTradingCurve::setSymbolExtent (double extent)

Set the extent of the symbol.

The width of the symbol is given in scale coordinates. When painting a symbol the width is scaled into paint device coordinates by scaledSymbolWidth(). The scaled width is bounded by minSymbolWidth(), maxSymbolWidth()

Parameters

| extent | Symbol width in scale coordinates |
|--------|-----------------------------------|

See Also

symbolExtent(), scaledSymbolWidth(), setMinSymbolWidth(), setMaxSymbolWidth()

12.94.4.19 void QwtPlotTradingCurve::setSymbolPen (const QColor & color, qreal width = 0.0, Qt::PenStyle style = Qt::SolidLine)

Build and assign the symbol pen

In Qt5 the default pen width is 1.0 (0.0 in Qt4) what makes it non cosmetic (see QPen::isCosmetic()). This method has been introduced to hide this incompatibility.

Parameters

| color | Pen color |
|-------|-----------|
| width | Pen width |
| style | Pen style |

See Also

pen(), brush()

12.94.4.20 void QwtPlotTradingCurve::setSymbolPen (const QPen & pen)

Set the symbol pen.

The symbol pen is used for rendering the lines of the bar or candlestick symbols

See Also

symbolPen(), setSymbolBrush()

12.94.4.21 void QwtPlotTradingCurve::setSymbolStyle (SymbolStyle style)

Set the symbol style

```
Parameters
```

```
style | Symbol style
```

See Also

symbolStyle(), setSymbolExtent(), setSymbolPen(), setSymbolBrush()

12.94.4.22 QBrush QwtPlotTradingCurve::symbolBrush (Direction direction) const

Parameters

direction

Returns

Brush used to fill the body of all candlestick symbols with the direction

See Also

setSymbolPen(), symbolBrush()

12.94.4.23 double QwtPlotTradingCurve::symbolExtent () const

Returns

Extent of a symbol in scale coordinates

See Also

setSymbolExtent(), scaledSymbolWidth(), minSymbolWidth(), maxSymbolWidth()

12.94.4.24 QPen QwtPlotTradingCurve::symbolPen () const

Returns

Symbol pen

See Also

setSymbolPen(), symbolBrush()

12.94.4.25 QwtPlotTradingCurve::SymbolStyle QwtPlotTradingCurve::symbolStyle () const

Returns

Symbol style

See Also

setSymbolStyle(), symbolExtent(), symbolPen(), symbolBrush()

12.94.4.26 bool QwtPlotTradingCurve::testPaintAttribute (PaintAttribute attribute) const

Returns

True, when attribute is enabled

See Also

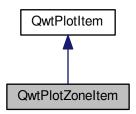
PaintAttribute, setPaintAttribute()

12.95 QwtPlotZoneItem Class Reference

A plot item, which displays a zone.

#include <qwt_plot_zoneitem.h>

Inheritance diagram for QwtPlotZoneItem:



Public Member Functions

QwtPlotZoneItem ()

Constructor.

virtual ~QwtPlotZoneItem ()

Destructor.

- · virtual int rtti () const
- void setOrientation (Qt::Orientation)

Set the orientation of the zone.

- Qt::Orientation orientation ()
- void setInterval (double min, double max)
- void setInterval (const QwtInterval &)
- · QwtInterval interval () const
- void setPen (const QColor &, greal width=0.0, Qt::PenStyle=Qt::SolidLine)
- void setPen (const QPen &)

Assign a pen.

- const QPen & pen () const
- void setBrush (const QBrush &)

Assign a brush.

- const QBrush & brush () const
- virtual void draw (QPainter *, const QwtScaleMap &, const QwtScaleMap &, const QRectF &) const
- virtual QRectF boundingRect () const

Additional Inherited Members

12.95.1 Detailed Description

A plot item, which displays a zone.

A horizontal zone highlights an interval of the y axis - a vertical zone an interval of the x axis - and is unbounded in the opposite direction. It is filled with a brush and its border lines are optionally displayed with a pen.

Note

For displaying an area that is bounded for x and y coordinates use QwtPlotShapeItem

12.95.2 Constructor & Destructor Documentation

12.95.2.1 QwtPlotZoneltem::QwtPlotZoneltem() [explicit]

Constructor.

Initializes the zone with no pen and a semi transparent gray brush

Sets the following item attributes:

· QwtPlotItem::AutoScale: false

QwtPlotItem::Legend: false

The z value is initialized by 5

See Also

QwtPlotItem::setItemAttribute(), QwtPlotItem::setZ()

12.95.3 Member Function Documentation

```
12.95.3.1 QRectF QwtPlotZoneItem::boundingRect() const [virtual]
```

The bounding rectangle is build from the interval in one direction and something invalid for the opposite direction.

Returns

An invalid rectangle with valid boundaries in one direction

Reimplemented from QwtPlotItem.

12.95.3.2 const QBrush & QwtPlotZoneItem::brush () const

Returns

Brush used to fill the zone

See Also

```
setPen(), brush()
```

12.95.3.3 void QwtPlotZoneltem::draw (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & canvasRect) const [virtual]

Draw the zone

Parameters

| painter | Painter |
|------------|---|
| хМар | x Scale Map |
| уМар | y Scale Map |
| canvasRect | Contents rectangle of the canvas in painter coordinates |

Implements QwtPlotItem.

12.95.3.4 QwtInterval QwtPlotZoneltem::interval () const

Returns

Zone interval

```
See Also
```

```
setInterval(), orientation()
```

12.95.3.5 Qt::Orientation QwtPlotZoneItem::orientation ()

Returns

Orientation of the zone

See Also

setOrientation()

12.95.3.6 const QPen & QwtPlotZoneItem::pen () const

Returns

Pen used to draw the border lines

See Also

```
setPen(), brush()
```

12.95.3.7 int QwtPlotZoneltem::rtti() const [virtual]

Returns

QwtPlotItem::Rtti_PlotZone

Reimplemented from QwtPlotItem.

12.95.3.8 void QwtPlotZoneltem::setBrush (const QBrush & brush)

Assign a brush.

The brush is used to fill the zone

Parameters

| brush | Brush |
|-------|-------|
|-------|-------|

See Also

pen(), setBrush()

12.95.3.9 void QwtPlotZoneltem::setInterval (double min, double max)

Set the interval of the zone

For a horizontal zone the interval is related to the y axis, for a vertical zone it is related to the x axis.

Parameters

| min | Minimum of the interval |
|-----|-------------------------|
| max | Maximum of the interval |

See Also

interval(), setOrientation()

12.95.3.10 void QwtPlotZoneltem::setInterval (const QwtInterval & interval)

Set the interval of the zone

For a horizontal zone the interval is related to the y axis, for a vertical zone it is related to the x axis.

| interval | Zone interval |
|----------|---------------|
|----------|---------------|

See Also

interval(), setOrientation()

12.95.3.11 void QwtPlotZoneItem::setOrientation (Qt::Orientation orientation)

Set the orientation of the zone.

A horizontal zone highlights an interval of the y axis, a vertical zone of the x axis. It is unbounded in the opposite direction.

See Also

orientation(), QwtPlotItem::setAxes()

12.95.3.12 void QwtPlotZoneltem::setPen (const QColor & color, qreal width = 0.0, Qt::PenStyle style = Qt::SolidLine)

Build and assign a pen

In Qt5 the default pen width is 1.0 (0.0 in Qt4) what makes it non cosmetic (see QPen::isCosmetic()). This method has been introduced to hide this incompatibility.

Parameters

| color | Pen color |
|-------|-----------|
| width | Pen width |
| style | Pen style |

See Also

pen(), brush()

12.95.3.13 void QwtPlotZoneltem::setPen (const QPen & pen)

Assign a pen.

The pen is used to draw the border lines of the zone

Parameters

| pen | Pen |
|-----|-----|
|-----|-----|

See Also

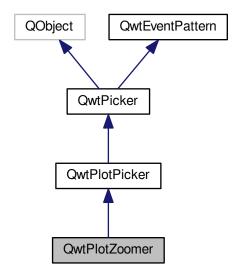
pen(), setBrush()

12.96 QwtPlotZoomer Class Reference

QwtPlotZoomer provides stacked zooming for a plot widget.

#include <qwt_plot_zoomer.h>

Inheritance diagram for QwtPlotZoomer:



Public Slots

- void moveBy (double x, double y)
- virtual void moveTo (const QPointF &)
- virtual void zoom (const QRectF &)

Zoom in.

• virtual void zoom (int up)

Zoom in or out.

Signals

void zoomed (const QRectF &rect)

Public Member Functions

QwtPlotZoomer (QWidget *, bool doReplot=true)

Create a zoomer for a plot canvas.

QwtPlotZoomer (int xAxis, int yAxis, QWidget *, bool doReplot=true)

Create a zoomer for a plot canvas.

- virtual void setZoomBase (bool doReplot=true)
- virtual void setZoomBase (const QRectF &)

Set the initial size of the zoomer.

- QRectF zoomBase () const
- QRectF zoomRect () const
- virtual void setAxis (int xAxis, int yAxis)
- void setMaxStackDepth (int)

Limit the number of recursive zoom operations to depth.

• int maxStackDepth () const

- const QStack< QRectF > & zoomStack () const
- void setZoomStack (const QStack< QRectF > &, int zoomRectIndex=-1)

Assign a zoom stack.

• uint zoomRectIndex () const

Protected Member Functions

- · virtual void rescale ()
- virtual QSizeF minZoomSize () const

Limit zooming by a minimum rectangle.

- virtual void widgetMouseReleaseEvent (QMouseEvent *)
- virtual void widgetKeyPressEvent (QKeyEvent *)
- virtual void begin ()
- virtual bool end (bool ok=true)
- virtual bool accept (QPolygon &) const

Check and correct a selected rectangle.

Additional Inherited Members

12.96.1 Detailed Description

QwtPlotZoomer provides stacked zooming for a plot widget.

QwtPlotZoomer selects rectangles from user inputs (mouse or keyboard) translates them into plot coordinates and adjusts the axes to them. The selection is supported by a rubber band and optionally by displaying the coordinates of the current mouse position.

Zooming can be repeated as often as possible, limited only by maxStackDepth() or minZoomSize(). Each rectangle is pushed on a stack.

The default setting how to select rectangles is a QwtPickerDragRectMachine with the following bindings:

QwtEventPattern::MouseSelect1

The first point of the zoom rectangle is selected by a mouse press, the second point from the position, where the mouse is released.

QwtEventPattern::KeySelect1

The first key press selects the first, the second key press selects the second point.

· QwtEventPattern::KeyAbort

Discard the selection in the state, where the first point is selected.

To traverse the zoom stack the following bindings are used:

QwtEventPattern::MouseSelect3, QwtEventPattern::KeyUndo

Zoom out one position on the zoom stack

• QwtEventPattern::MouseSelect6, QwtEventPattern::KeyRedo

Zoom in one position on the zoom stack

· QwtEventPattern::MouseSelect2, QwtEventPattern::KeyHome

Zoom to the zoom base

The setKeyPattern() and setMousePattern() functions can be used to configure the zoomer actions. The following example shows, how to configure the 'l' and 'O' keys for zooming in and out one position on the zoom stack. The "Home" key is used to "unzoom" the plot.

```
zoomer = new QwtPlotZoomer( plot );
zoomer->setKeyPattern( QwtEventPattern::KeyRedo, Qt::Key_I, Qt::ShiftModifier );
zoomer->setKeyPattern( QwtEventPattern::KeyUndo, Qt::Key_O, Qt::ShiftModifier );
zoomer->setKeyPattern( QwtEventPattern::KeyHome, Qt::Key_Home );
```

QwtPlotZoomer is tailored for plots with one x and y axis, but it is allowed to attach a second QwtPlotZoomer (without rubber band and tracker) for the other axes.

Note

The realtime example includes an derived zoomer class that adds scrollbars to the plot canvas.

See Also

QwtPlotPanner, QwtPlotMagnifier

12.96.2 Constructor & Destructor Documentation

```
12.96.2.1 QwtPlotZoomer::QwtPlotZoomer ( QWidget * canvas, bool doReplot = true ) [explicit]
```

Create a zoomer for a plot canvas.

The zoomer is set to those x- and y-axis of the parent plot of the canvas that are enabled. If both or no x-axis are enabled, the picker is set to QwtPlot::xBottom. If both or no y-axis are enabled, it is set to QwtPlot::yLeft.

The zoomer is initialized with a QwtPickerDragRectMachine, the tracker mode is set to QwtPicker::ActiveOnly and the rubber band is set to QwtPicker::RectRubberBand

Parameters

| canvas | Plot canvas to observe, also the parent object |
|----------|---|
| doReplot | Call QwtPlot::replot() for the attached plot before initializing the zoomer with its scales. This |
| | might be necessary, when the plot is in a state with pending scale changes. |

See Also

QwtPlot::autoReplot(), QwtPlot::replot(), setZoomBase()

```
12.96.2.2 QwtPlotZoomer::QwtPlotZoomer( int xAxis, int yAxis, QWidget * canvas, bool doReplot = true )
[explicit]
```

Create a zoomer for a plot canvas.

The zoomer is initialized with a QwtPickerDragRectMachine, the tracker mode is set to QwtPicker::ActiveOnly and the rubber band is set to QwtPicker;;RectRubberBand

Parameters

| xAxis | X axis of the zoomer |
|----------|---|
| yAxis | Y axis of the zoomer |
| canvas | Plot canvas to observe, also the parent object |
| doReplot | Call QwtPlot::replot() for the attached plot before initializing the zoomer with its scales. This |
| | might be necessary, when the plot is in a state with pending scale changes. |

See Also

QwtPlot::autoReplot(), QwtPlot::replot(), setZoomBase()

12.96.3 Member Function Documentation

```
12.96 QwtPlotZoomer Class Reference
                                                                                                          481
12.96.3.1 bool QwtPlotZoomer::accept ( QPolygon & pa ) const [protected], [virtual]
Check and correct a selected rectangle.
Reject rectangles with a height or width < 2, otherwise expand the selected rectangle to a minimum size of 11x11
and accept it.
Returns
     true If the rectangle is accepted, or has been changed to an accepted one.
Reimplemented from QwtPicker.
12.96.3.2 void QwtPlotZoomer::begin ( ) [protected], [virtual]
Rejects selections, when the stack depth is too deep, or the zoomed rectangle is minZoomSize().
See Also
     minZoomSize(), maxStackDepth()
Reimplemented from QwtPicker.
12.96.3.3 bool QwtPlotZoomer::end (bool ok = true ) [protected], [virtual]
Expand the selected rectangle to \mbox{minZoomSize}() and zoom in if accepted.
Parameters
                    If true, complete the selection and emit selected signals otherwise discard the selection.
See Also
     accept(), minZoomSize()
Returns
     True if the selection has been accepted, false otherwise
Reimplemented from QwtPlotPicker.
12.96.3.4 int QwtPlotZoomer::maxStackDepth ( ) const
Returns
     Maximal depth of the zoom stack.
See Also
     setMaxStackDepth()
12.96.3.5 QSizeF QwtPlotZoomer::minZoomSize() const [protected], [virtual]
```

Returns

```
zoomBase().width() / 10e4, zoomBase().height() / 10e4
```

12.96.3.6 void QwtPlotZoomer::moveBy (double dx, double dy) [slot]

Move the current zoom rectangle.

Limit zooming by a minimum rectangle.

Parameters

| dx | X offset |
|----|----------|
| dy | Y offset |

Note

The changed rectangle is limited by the zoom base

12.96.3.7 void QwtPlotZoomer::moveTo (const QPointF & pos) [virtual], [slot]

Move the the current zoom rectangle.

Parameters

| pos | New position |
|-----|--------------|

See Also

QRectF::moveTo()

Note

The changed rectangle is limited by the zoom base

12.96.3.8 void QwtPlotZoomer::rescale() [protected], [virtual]

Adjust the observed plot to zoomRect()

Note

Initiates QwtPlot::replot()

12.96.3.9 void QwtPlotZoomer::setAxis(int xAxis, int yAxis) [virtual]

Reinitialize the axes, and set the zoom base to their scales.

Parameters

| xAxis | X axis |
|-------|--------|
| yAxis | Y axis |

Reimplemented from QwtPlotPicker.

12.96.3.10 void QwtPlotZoomer::setMaxStackDepth (int depth)

Limit the number of recursive zoom operations to depth.

A value of -1 set the depth to unlimited, 0 disables zooming. If the current zoom rectangle is below depth, the plot is unzoomed.

Parameters

| depth | Maximum for the stack depth |
|-------|-----------------------------|
|-------|-----------------------------|

See Also

maxStackDepth()

Note

depth doesn't include the zoom base, so zoomStack().count() might be maxStackDepth() + 1.

12.96.3.11 void QwtPlotZoomer::setZoomBase (bool doReplot = true) [virtual]

Reinitialized the zoom stack with scaleRect() as base.

Parameters

| doReplot | Call QwtPlot::replot() for the attached plot before initializing the zoomer with its scales. This |
|----------|---|
| | might be necessary, when the plot is in a state with pending scale changes. |

See Also

zoomBase(), scaleRect() QwtPlot::autoReplot(), QwtPlot::replot().

12.96.3.12 void QwtPlotZoomer::setZoomBase(const QRectF & base) [virtual]

Set the initial size of the zoomer.

base is united with the current scaleRect() and the zoom stack is reinitialized with it as zoom base. plot is zoomed to scaleRect().

Parameters

| base | Zoom base |
|------|-----------|
|------|-----------|

See Also

zoomBase(), scaleRect()

12.96.3.13 void QwtPlotZoomer::setZoomStack (const QStack < QRectF > & zoomStack, int zoomRectIndex = -1)

Assign a zoom stack.

In combination with other types of navigation it might be useful to modify to manipulate the complete zoom stack.

Parameters

| zoomStack | New zoom stack |
|---------------|--|
| zoomRectIndex | Index of the current position of zoom stack. In case of -1 the current position is at the top of |
| | the stack. |

Note

The zoomed signal might be emitted.

See Also

zoomStack(), zoomRectIndex()

12.96.3.14 void QwtPlotZoomer::widgetKeyPressEvent (QKeyEvent * ke) [protected], [virtual]

Qt::Key_Plus zooms in, Qt::Key_Minus zooms out one position on the zoom stack, Qt::Key_Escape zooms out to the zoom base.

Changes the current position on the stack, but doesn't pop any rectangle.

Note

The keys codes can be changed, using QwtEventPattern::setKeyPattern: 3, 4, 5

Reimplemented from QwtPicker.

12.96.3.15 void QwtPlotZoomer::widgetMouseReleaseEvent (QMouseEvent * me) [protected], [virtual]

Qt::MidButton zooms out one position on the zoom stack, Qt::RightButton to the zoom base.

Changes the current position on the stack, but doesn't pop any rectangle.

Note

The mouse events can be changed, using QwtEventPattern::setMousePattern: 2, 1

Reimplemented from QwtPicker.

```
12.96.3.16 void QwtPlotZoomer::zoom ( const QRectF & rect ) [virtual], [slot]
```

Zoom in.

Clears all rectangles above the current position of the zoom stack and pushes the normalized rectangle on it.

Note

If the maximal stack depth is reached, zoom is ignored.

The zoomed signal is emitted.

```
12.96.3.17 void QwtPlotZoomer::zoom (int offset) [virtual], [slot]
```

Zoom in or out.

Activate a rectangle on the zoom stack with an offset relative to the current position. Negative values of offset will zoom out, positive zoom in. A value of 0 zooms out to the zoom base.

Parameters

offset Offset relative to the current position of the zoom stack.

Note

The zoomed signal is emitted.

See Also

zoomRectIndex()

12.96.3.18 QRectF QwtPlotZoomer::zoomBase () const

Returns

Initial rectangle of the zoomer

See Also

```
setZoomBase(), zoomRect()
```

```
12.96.3.19 void QwtPlotZoomer::zoomed ( const QRectF & rect ) [signal]
```

A signal emitting the zoomRect(), when the plot has been zoomed in or out.

Parameters

rect Current zoom rectangle.

12.96.3.20 QRectF QwtPlotZoomer::zoomRect () const

Returns

Rectangle at the current position on the zoom stack.

See Also

zoomRectIndex(), scaleRect().

```
12.96.3.21 uint QwtPlotZoomer::zoomRectIndex ( ) const
```

Returns

Index of current position of zoom stack.

```
12.96.3.22 const QStack< QRectF > & QwtPlotZoomer::zoomStack ( ) const
```

Returns

The zoom stack. zoomStack()[0] is the zoom base, zoomStack()[1] the first zoomed rectangle.

See Also

```
setZoomStack(), zoomRectIndex()
```

12.97 QwtPoint3D Class Reference

QwtPoint3D class defines a 3D point in double coordinates.

```
#include <qwt_point_3d.h>
```

Public Member Functions

- · QwtPoint3D ()
- QwtPoint3D (double x, double y, double z)

Constructs a point with coordinates specified by x, y and z.

- QwtPoint3D (const QwtPoint3D &)
- QwtPoint3D (const QPointF &)
- bool isNull () const
- · double x () const
- · double y () const
- double z () const
- · double & rx ()
- double & ry ()
- double & rz ()
- void setX (double x)

Sets the x-coordinate of the point to the value specified by x.

void setY (double y)

Sets the y-coordinate of the point to the value specified by y.

void setZ (double y)

Sets the z-coordinate of the point to the value specified by z.

- QPointF toPoint () const
- bool operator== (const QwtPoint3D &) const
- bool operator!= (const QwtPoint3D &) const

12.97.1 Detailed Description

QwtPoint3D class defines a 3D point in double coordinates.

```
12.97.2 Constructor & Destructor Documentation
12.97.2.1 QwtPoint3D::QwtPoint3D( ) [inline]
Constructs a null point.
See Also
      isNull()
12.97.2.2 QwtPoint3D::QwtPoint3D ( const QwtPoint3D & other ) [inline]
Copy constructor. Constructs a point using the values of the point specified.
12.97.2.3 QwtPoint3D::QwtPoint3D (const QPointF & other) [inline]
Constructs a point with x and y coordinates from a 2D point, and a z coordinate of 0.
12.97.3 Member Function Documentation
12.97.3.1 bool QwtPoint3D::isNull() const [inline]
Returns
      True if the point is null; otherwise returns false.
A point is considered to be null if x, y and z-coordinates are equal to zero.
12.97.3.2 bool QwtPoint3D::operator!= ( const QwtPoint3D & other ) const [inline]
Returns
      True if this rect and other are different; otherwise returns false.
12.97.3.3 bool QwtPoint3D::operator== ( const QwtPoint3D & other ) const [inline]
Returns
      True, if this point and other are equal; otherwise returns false.
12.97.3.4 double & QwtPoint3D::rx( ) [inline]
Returns
      A reference to the x-coordinate of the point.
12.97.3.5 double & QwtPoint3D::ry() [inline]
Returns
      A reference to the y-coordinate of the point.
12.97.3.6 double & QwtPoint3D::rz( ) [inline]
Returns
      A reference to the z-coordinate of the point.
```

12.97.3.7 QPointF QwtPoint3D::toPoint() const [inline]

Returns

2D point, where the z coordinate is dropped.

12.97.3.8 double QwtPoint3D::x() const [inline]

Returns

The x-coordinate of the point.

12.97.3.9 double QwtPoint3D::y() const [inline]

Returns

The y-coordinate of the point.

12.97.3.10 double QwtPoint3D::z() const [inline]

Returns

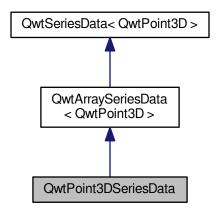
The z-coordinate of the point.

12.98 QwtPoint3DSeriesData Class Reference

Interface for iterating over an array of 3D points.

#include <qwt_series_data.h>

Inheritance diagram for QwtPoint3DSeriesData:



Public Member Functions

- QwtPoint3DSeriesData (const QVector< QwtPoint3D > &=QVector< QwtPoint3D >())
- · virtual QRectF boundingRect () const

Calculate the bounding rectangle.

Additional Inherited Members

12.98.1 Detailed Description

Interface for iterating over an array of 3D points.

12.98.2 Constructor & Destructor Documentation

12.98.2.1 QwtPoint3DSeriesData::QwtPoint3DSeriesData (const QVector< QwtPoint3D > & samples = QVector<QwtPoint3D>())

Constructor

Parameters

samples Samples

12.98.3 Member Function Documentation

12.98.3.1 QRectF QwtPoint3DSeriesData::boundingRect() const [virtual]

Calculate the bounding rectangle.

The bounding rectangle is calculated once by iterating over all points and is stored for all following requests.

Returns

Bounding rectangle

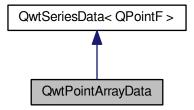
Implements QwtSeriesData < QwtPoint3D >.

12.99 QwtPointArrayData Class Reference

Interface for iterating over two QVector<double> objects.

#include <qwt_point_data.h>

Inheritance diagram for QwtPointArrayData:



Public Member Functions

- QwtPointArrayData (const QVector< double > &x, const QVector< double > &y)
- QwtPointArrayData (const double *x, const double *y, size_t size)

• virtual QRectF boundingRect () const

Calculate the bounding rectangle.

- virtual size_t size () const
- virtual QPointF sample (size t i) const
- const QVector< double > & xData () const
- const QVector< double > & yData () const

Additional Inherited Members

12.99.1 Detailed Description

Interface for iterating over two QVector<double> objects.

12.99.2 Constructor & Destructor Documentation

12.99.2.1 QwtPointArrayData::QwtPointArrayData (const QVector< double > & x, const QVector< double > & y)

Constructor

Parameters

| X | Array of x values |
|---|-------------------|
| у | Array of y values |

See Also

QwtPlotCurve::setData(), QwtPlotCurve::setSamples()

12.99.2.2 QwtPointArrayData::QwtPointArrayData (const double * x, const double * y, size_t size)

Constructor

Parameters

| X | Array of x values |
|------|----------------------------|
| У | Array of y values |
| size | Size of the x and y arrays |

See Also

QwtPlotCurve::setData(), QwtPlotCurve::setSamples()

12.99.3 Member Function Documentation

12.99.3.1 QRectF QwtPointArrayData::boundingRect() const [virtual]

Calculate the bounding rectangle.

The bounding rectangle is calculated once by iterating over all points and is stored for all following requests.

Returns

Bounding rectangle

Implements QwtSeriesData < QPointF >.

12.99.3.2 QPointF QwtPointArrayData::sample (size_t index) const [virtual]

Return the sample at position i

Parameters

| index | Index |
|-------|-------|

Returns

Sample at position i

Implements QwtSeriesData < QPointF >.

12.99.3.3 size_t QwtPointArrayData::size() const [virtual]

Returns

Size of the data set

Implements QwtSeriesData < QPointF >.

12.99.3.4 const QVector< double > & QwtPointArrayData::xData () const

Returns

Array of the x-values

12.99.3.5 const QVector < double > & QwtPointArrayData::yData () const

Returns

Array of the y-values

12.100 QwtPointMapper Class Reference

A helper class for translating a series of points.

```
#include <qwt_point_mapper.h>
```

Public Types

enum TransformationFlag { RoundPoints = 0x01, WeedOutPoints = 0x02 }

Flags affecting the transformation process.

- · typedef QFlags
 - < TransformationFlag > TransformationFlags

Flags affecting the transformation process.

Public Member Functions

QwtPointMapper ()

Constructor.

∼QwtPointMapper ()

Destructor.

- void setFlags (TransformationFlags)
- · TransformationFlags flags () const
- void setFlag (TransformationFlag, bool on=true)
- bool testFlag (TransformationFlag) const
- void setBoundingRect (const QRectF &)
- QRectF boundingRect () const
- QPolygonF toPolygonF (const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QwtSeriesData
 QPointF > *series, int from, int to) const

Translate a series of points into a QPolygonF.

QPolygon toPolygon (const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QwtSeriesData < Q-PointF > *series, int from, int to) const

Translate a series of points into a QPolygon.

 QPolygon toPoints (const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QwtSeriesData < QPoint-F > *series, int from, int to) const

Translate a series of points into a QPolygon.

QPolygonF toPointsF (const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QwtSeriesData < Q-PointF > *series, int from, int to) const

Translate a series into a QPolygonF.

QImage toImage (const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QwtSeriesData < QPointF > *series, int from, int to, const QPen &, bool antialiased, uint numThreads) const

Translate a series into a QImage.

12.100.1 Detailed Description

A helper class for translating a series of points.

QwtPointMapper is a collection of methods and optimizations for translating a series of points into paint device coordinates. It is used by QwtPlotCurve but might also be useful for similar plot items displaying a QwtSeries-Data<QPointF>.

12.100.2 Member Typedef Documentation

12.100.2.1 typedef QFlags<TransformationFlag> QwtPointMapper::TransformationFlags

Flags affecting the transformation process.

See Also

```
setFlag(), setFlags()
```

12.100.3 Member Enumeration Documentation

12.100.3.1 enum QwtPointMapper::TransformationFlag

Flags affecting the transformation process.

See Also

```
setFlag(), setFlags()
```

Enumerator

RoundPoints Round points to integer values.

WeedOutPoints Try to remove points, that are translated to the same position.

12.100.4 Member Function Documentation

12.100.4.1 QRectF QwtPointMapper::boundingRect () const

Returns

Bounding rectangle

See Also

setBoundingRect()

12.100.4.2 QwtPointMapper::TransformationFlags QwtPointMapper::flags () const

Returns

Flags affecting the transformation process

See Also

```
setFlags(), setFlag()
```

12.100.4.3 void QwtPointMapper::setBoundingRect (const QRectF & rect)

Set a bounding rectangle for the point mapping algorithm

A valid bounding rectangle can be used for optimizations

Parameters

| rect | Bounding rectangle |
|------|--------------------|

See Also

boundingRect()

12.100.4.4 void QwtPointMapper::setFlag (TransformationFlag flag, bool on = true)

Modify a flag affecting the transformation process

Parameters

| flag | Flag type |
|------|-----------|
| on | Value |

See Also

flag(), setFlags()

12.100.4.5 void QwtPointMapper::setFlags (TransformationFlags flags)

Set the flags affecting the transformation process

Parameters

| flags Flags | |
|-------------|--|
|-------------|--|

See Also

flags(), setFlag()

12.100.4.6 bool QwtPointMapper::testFlag (TransformationFlag flag) const

Returns

True, when the flag is set

Parameters

| flag | Flag type |
|------|-----------|
|------|-----------|

See Also

setFlag(), setFlags()

12.100.4.7 QImage QwtPointMapper::toImage (const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QwtSeriesData < QPointF > * series, int from, int to, const QPen & pen, bool antialiased, uint numThreads) const

Translate a series into a QImage.

Parameters

| хМар | х тар |
|-------------|--|
| уМар | y map |
| series | Series of points to be mapped |
| from | Index of the first point to be painted |
| to | Index of the last point to be painted |
| pen | Pen used for drawing a point of the image, where a point is mapped to |
| antialiased | True, when the dots should be displayed antialiased |
| numThreads | Number of threads to be used for rendering. If numThreads is set to 0, the system specific |
| | ideal thread count is used. |

Returns

Image displaying the series

12.100.4.8 QPolygon QwtPointMapper::toPoints (const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QwtSeriesData < QPointF > * series, int from, int to) const

Translate a series of points into a QPolygon.

- WeedOutPoints & boundingRect().isValid() All points that are mapped to the same position will be one point. Points outside of the bounding rectangle are ignored.
- WeedOutPoints & !boundingRect().isValid() All consecutive points that are mapped to the same position will one point
- !WeedOutPoints & boundingRect().isValid() Points outside of the bounding rectangle are ignored.

Parameters

| хМар | х тар |
|--------|--|
| уМар | y map |
| series | Series of points to be mapped |
| from | Index of the first point to be painted |
| to | Index of the last point to be painted |

Returns

Translated polygon

12.100.4.9 QPolygonF QwtPointMapper::toPointsF (const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QwtSeriesData < QPointF > * series, int from, int to) const

Translate a series into a QPolygonF.

- WeedOutPoints & RoundPoints & boundingRect().isValid() All points that are mapped to the same position will be one point. Points outside of the bounding rectangle are ignored.
- WeedOutPoints & RoundPoints & !boundingRect().isValid() All consecutive points that are mapped to the same position will one point
- · WeedOutPoints & !RoundPoints All consecutive points that are mapped to the same position will one point
- !WeedOutPoints & boundingRect().isValid() Points outside of the bounding rectangle are ignored.

When RoundPoints is set all points are rounded to integers but returned as PolygonF - what only makes sense when the further processing of the values need a QPolygonF.

Parameters

| хМар | х тар |
|--------|--|
| уМар | y map |
| series | Series of points to be mapped |
| from | Index of the first point to be painted |
| to | Index of the last point to be painted |

Returns

Translated polygon

12.100.4.10 QPolygon QwtPointMapper::toPolygon (const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QwtSeriesData< QPointF > * series, int from, int to) const

Translate a series of points into a QPolygon.

When the WeedOutPoints flag is enabled consecutive points, that are mapped to the same position will be one point.

Parameters

| хМар | х тар |
|--------|--|
| уМар | y map |
| series | Series of points to be mapped |
| from | Index of the first point to be painted |
| to | Index of the last point to be painted |

Returns

Translated polygon

12.100.4.11 QPolygonF QwtPointMapper::toPolygonF (const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QwtSeriesData< QPointF > * series, int from, int to) const

Translate a series of points into a QPolygonF.

When the WeedOutPoints flag is enabled consecutive points, that are mapped to the same position will be one point.

When RoundPoints is set all points are rounded to integers but returned as PolygonF - what only makes sense when the further processing of the values need a QPolygonF.

Parameters

| хМар | x map |
|--------|--|
| уМар | y map |
| series | Series of points to be mapped |
| from | Index of the first point to be painted |
| to | Index of the last point to be painted |

Returns

Translated polygon

12.101 QwtPointPolar Class Reference

A point in polar coordinates.

```
#include <qwt_point_polar.h>
```

Public Member Functions

- · QwtPointPolar ()
- · QwtPointPolar (double azimuth, double radius)
- QwtPointPolar (const QwtPointPolar &)
- QwtPointPolar (const QPointF &)
- void setPoint (const QPointF &)
- QPointF toPoint () const
- bool isValid () const

Returns true if radius() >= 0.0.

• bool isNull () const

Returns true if radius() >= 0.0.

· double radius () const

Returns the radius.

• double azimuth () const

Returns the azimuth.

• double & rRadius ()

Returns the radius.

double & rAzimuth ()

Returns the azimuth.

• void setRadius (double)

Sets the radius to radius.

void setAzimuth (double)

Sets the atimuth to atimuth.

bool operator== (const QwtPointPolar &) const

Compare 2 points.

- bool operator!= (const QwtPointPolar &) const
- · QwtPointPolar normalized () const

12.101.1 Detailed Description

A point in polar coordinates.

In polar coordinates a point is determined by an angle and a distance. See http://en.wikipedia.-org/wiki/Polar_coordinate_system

12.101.2 Constructor & Destructor Documentation

12.101.2.1 QwtPointPolar::QwtPointPolar() [inline]

Constructs a null point, with a radius and azimuth set to 0.0.

See Also

QPointF::isNull()

12.101.2.2 QwtPointPolar::QwtPointPolar (double azimuth, double radius) [inline]

Constructs a point with coordinates specified by radius and azimuth.

Parameters

| azimuth | Azimuth |
|---------|---------|
| radius | Radius |

12.101.2.3 QwtPointPolar::QwtPointPolar (const QwtPointPolar & other) [inline]

Constructs a point using the values of the point specified.

Parameters

| other | Other point |
|-------|-------------|

12.101.2.4 QwtPointPolar::QwtPointPolar (const QPointF & p)

Convert and assign values from a point in Cartesian coordinates

Parameters

| , , , , , , , , , , , , , , , , , , , | р | Point in Cartesian coordinates |
|---------------------------------------|---|--------------------------------|
|---------------------------------------|---|--------------------------------|

See Also

setPoint(), toPoint()

12.101.3 Member Function Documentation

 $12.101.3.1 \quad \textbf{QwtPointPolar} \ \textbf{QwtPointPolar::normalized (} \quad \textbf{)} \ \textbf{const}$

Normalize radius and azimuth

When the radius is < 0.0 it is set to 0.0. The azimuth is a value >= 0.0 and < 2 * M_PI.

Returns

Normalized point

12.101.3.2 bool QwtPointPolar::operator!= (const QwtPointPolar & other) const

Compare 2 points

Two points are equal to each other if radius and azimuth-coordinates are the same. Points are not equal, when the azimuth differs, but other.azimuth() == azimuth() % (2 * PI).

Returns

True if the point is not equal to other; otherwise return false.

| 200 | ٨ | اوم | |
|-----|---|-----|--|
| | | | |

normalized()

12.101.3.3 bool QwtPointPolar::operator== (const QwtPointPolar & other) const

Compare 2 points.

Two points are equal to each other if radius and azimuth-coordinates are the same. Points are not equal, when the azimuth differs, but other.azimuth() == azimuth() % (2 * PI).

Returns

True if the point is equal to other; otherwise return false.

See Also

normalized()

12.101.3.4 void QwtPointPolar::setPoint (const QPointF & p)

Convert and assign values from a point in Cartesian coordinates

Parameters

p Point in Cartesian coordinates

12.101.3.5 QPointF QwtPointPolar::toPoint () const

Convert and return values in Cartesian coordinates

Returns

Converted point in Cartesian coordinates

Note

Invalid or null points will be returned as QPointF(0.0, 0.0)

See Also

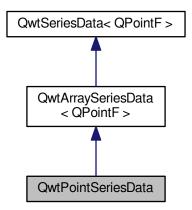
isValid(), isNull()

12.102 QwtPointSeriesData Class Reference

Interface for iterating over an array of points.

```
#include <qwt_series_data.h>
```

Inheritance diagram for QwtPointSeriesData:



Public Member Functions

- QwtPointSeriesData (const QVector< QPointF > &=QVector< QPointF >())
- virtual QRectF boundingRect () const

Calculate the bounding rectangle.

Additional Inherited Members

12.102.1 Detailed Description

Interface for iterating over an array of points.

12.102.2 Constructor & Destructor Documentation

12.102.2.1 QwtPointSeriesData::QwtPointSeriesData (const QVector < QPointF > & samples = QVector < QPointF > ()

Constructor

Parameters

samples Samples

12.102.3 Member Function Documentation

12.102.3.1 QRectF QwtPointSeriesData::boundingRect() const [virtual]

Calculate the bounding rectangle.

The bounding rectangle is calculated once by iterating over all points and is stored for all following requests.

Returns

Bounding rectangle

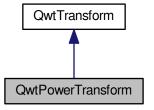
Implements QwtSeriesData < QPointF >.

12.103 QwtPowerTransform Class Reference

A transformation using pow()

#include <qwt_transform.h>

Inheritance diagram for QwtPowerTransform:



Public Member Functions

- · QwtPowerTransform (double exponent)
- virtual \sim QwtPowerTransform ()

Destructor.

- · virtual double transform (double value) const
- virtual double invTransform (double value) const
- virtual QwtTransform * copy () const

12.103.1 Detailed Description

A transformation using pow()

QwtPowerTransform preserves the sign of a value. F.e. a transformation with a factor of 2 transforms a value of -3 to -9 and v.v. Thus QwtPowerTransform can be used for scales including negative values.

12.103.2 Constructor & Destructor Documentation

12.103.2.1 QwtPowerTransform::QwtPowerTransform (double exponent)

Constructor

Parameters

| exponent | Exponent |
|----------|----------|

12.103.3 Member Function Documentation

12.103.3.1 QwtTransform * QwtPowerTransform::copy() const [virtual]

Returns

Clone of the transformation

Implements QwtTransform.

12.103.3.2 double QwtPowerTransform::invTransform (double value) const [virtual]

Parameters

value Value to be transformed

Returns

Inverse exponentiation preserving the sign

Implements QwtTransform.

12.103.3.3 double QwtPowerTransform::transform (double value) const [virtual]

Parameters

| value | Value to be transformed |
|-------|-------------------------|

Returns

Exponentiation preserving the sign

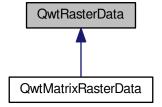
Implements QwtTransform.

12.104 QwtRasterData Class Reference

QwtRasterData defines an interface to any type of raster data.

#include <qwt_raster_data.h>

Inheritance diagram for QwtRasterData:



Public Types

enum ConrecFlag { IgnoreAllVerticesOnLevel = 0x01, IgnoreOutOfRange = 0x02 }

Flags to modify the contour algorithm.

typedef QMap< double, QPolygonF > ContourLines

Contour lines.

typedef QFlags < ConrecFlags

Flags to modify the contour algorithm.

Public Member Functions

· QwtRasterData ()

Constructor.

virtual ~QwtRasterData ()

Destructor.

- virtual void setInterval (Qt::Axis, const QwtInterval &)
- · const QwtInterval & interval (Qt::Axis) const
- virtual QRectF pixelHint (const QRectF &) const

Pixel him

• virtual void initRaster (const QRectF &, const QSize &raster)

Initialize a raster.

virtual void discardRaster ()

Discard a raster.

- virtual double value (double x, double y) const =0
- virtual ContourLines contourLines (const QRectF &rect, const QSize &raster, const QList< double > &levels,
 ConrecFlags) const

12.104.1 Detailed Description

QwtRasterData defines an interface to any type of raster data.

QwtRasterData is an abstract interface, that is used by QwtPlotRasterItem to find the values at the pixels of its raster.

Often a raster item is used to display values from a matrix. Then the derived raster data class needs to implement some sort of resampling, that maps the raster of the matrix into the requested raster of the raster item (depending on resolution and scales of the canvas).

12.104.2 Member Enumeration Documentation

12.104.2.1 enum QwtRasterData::ConrecFlag

Flags to modify the contour algorithm.

Enumerator

IgnoreAllVerticesOnLevel Ignore all vertices on the same level. *IgnoreOutOfRange* Ignore all values, that are out of range.

12.104.3 Member Function Documentation

12.104.3.1 QwtRasterData::ContourLines QwtRasterData::contourLines (const QRectF & rect, const QSize & raster, const QList< double > & levels, ConrecFlags flags) const [virtual]

Calculate contour lines

Parameters

| rect | Bounding rectangle for the contour lines |
|--------|---|
| raster | Number of data pixels of the raster data |
| levels | List of limits, where to insert contour lines |
| flags | Flags to customize the contouring algorithm |

Returns

Calculated contour lines

An adaption of CONREC, a simple contouring algorithm. http://local.wasp.uwa.edu.au/~pbourke/papers/conre

12.104.3.2 void QwtRasterData::discardRaster() [virtual]

Discard a raster.

After the composition of an image QwtPlotSpectrogram calls discardRaster().

The default implementation does nothing, but if data has been loaded in initRaster(), it could deleted now.

See Also

initRaster(), value()

12.104.3.3 void QwtRasterData::initRaster (const QRectF & area, const QSize & raster) [virtual]

Initialize a raster.

Before the composition of an image QwtPlotSpectrogram calls initRaster(), announcing the area and its resolution that will be requested.

The default implementation does nothing, but for data sets that are stored in files, it might be good idea to reimplement initRaster(), where the data is resampled and loaded into memory.

Parameters

| area | Area of the raster |
|--------|--|
| raster | Number of horizontal and vertical pixels |

See Also

initRaster(), value()

12.104.3.4 const QwtInterval & QwtRasterData::interval (Qt::Axis axis) const [inline]

Returns

Bounding interval for a axis

See Also

setInterval

12.104.3.5 QRectF QwtRasterData::pixelHint (const QRectF & area) const [virtual]

Pixel hint.

pixelHint() returns the geometry of a pixel, that can be used to calculate the resolution and alignment of the plot item, that is representing the data.

Width and height of the hint need to be the horizontal and vertical distances between 2 neighbored points. The center of the hint has to be the position of any point (it doesn't matter which one).

An empty hint indicates, that there are values for any detail level.

Limiting the resolution of the image might significantly improve the performance and heavily reduce the amount of memory when rendering a QImage from the raster data.

The default implementation returns an empty rectangle recommending to render in target device (f.e. screen) resolution.

Parameters

| area | In most implementations the resolution of the data doesn't depend on the requested area. |
|------|--|
|------|--|

Returns

Bounding rectangle of a pixel

Reimplemented in QwtMatrixRasterData.

12.104.3.6 void QwtRasterData::setInterval (Qt::Axis axis, const QwtInterval & interval) [virtual]

Set the bounding interval for the x, y or z coordinates.

Parameters

| axis | Axis |
|----------|-------------------|
| interval | Bounding interval |

See Also

interval()

Reimplemented in QwtMatrixRasterData.

12.104.3.7 virtual double QwtRasterData::value (double x, double y) const [pure virtual]

Returns

the value at a raster position

Parameters

| Х | X value in plot coordinates |
|---|-----------------------------|
| У | Y value in plot coordinates |

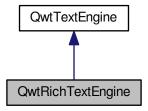
Implemented in QwtMatrixRasterData.

12.105 QwtRichTextEngine Class Reference

A text engine for Qt rich texts.

#include <qwt_text_engine.h>

Inheritance diagram for QwtRichTextEngine:



Public Member Functions

• QwtRichTextEngine ()

Constructor.

- · virtual double heightForWidth (const QFont &font, int flags, const QString &text, double width) const
- virtual QSizeF textSize (const QFont &font, int flags, const QString &text) const
- virtual void draw (QPainter *painter, const QRectF &rect, int flags, const QString &text) const
- virtual bool mightRender (const QString &) const
- virtual void textMargins (const QFont &, const QString &, double &left, double &right, double &top, double &bottom) const

Additional Inherited Members

12.105.1 Detailed Description

A text engine for Qt rich texts.

QwtRichTextEngine renders Qt rich texts using the classes of the Scribe framework of Qt.

12.105.2 Member Function Documentation

12.105.2.1 void QwtRichTextEngine::draw (QPainter * painter, const QRectF & rect, int flags, const QString & text) const [virtual]

Draw the text in a clipping rectangle

Parameters

| painter | Painter |
|---------|--|
| rect | Clipping rectangle |
| flags | Bitwise OR of the flags like in for QPainter::drawText() |
| text | Text to be rendered |

Implements QwtTextEngine.

12.105.2.2 double QwtRichTextEngine::heightForWidth (const QFont & font, int flags, const QString & text, double width) const [virtual]

Find the height for a given width

Parameters

| font | Font of the text |
|-------|---|
| flags | Bitwise OR of the flags used like in QPainter::drawText() |
| text | Text to be rendered |
| width | Width |

Returns

Calculated height

Implements QwtTextEngine.

12.105.2.3 bool QwtRichTextEngine::mightRender (const QString & text) const [virtual]

Test if a string can be rendered by this text engine

Parameters

| text | Text to be tested |
|------|-------------------|

Returns

Qt::mightBeRichText(text);

Implements QwtTextEngine.

12.105.2.4 void QwtRichTextEngine::textMargins (const QFont & , const QString & , double & *left*, double & *right*, double & *top*, double & *bottom*) const [virtual]

Return margins around the texts

Parameters

| left | Return 0 |
|--------|----------|
| right | Return 0 |
| top | Return 0 |
| bottom | Return 0 |

Implements QwtTextEngine.

12.105.2.5 QSizeF QwtRichTextEngine::textSize (const QFont & font, int flags, const QString & text) const [virtual]

Returns the size, that is needed to render text

Parameters

| font | Font of the text |
|-------|---|
| flags | Bitwise OR of the flags used like in QPainter::drawText() |
| text | Text to be rendered |

Returns

Caluclated size

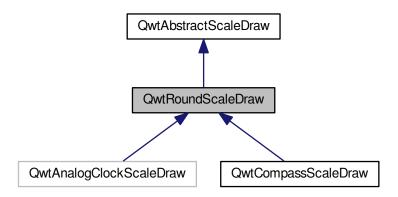
Implements QwtTextEngine.

12.106 QwtRoundScaleDraw Class Reference

A class for drawing round scales.

#include <qwt_round_scale_draw.h>

Inheritance diagram for QwtRoundScaleDraw:



Public Member Functions

• QwtRoundScaleDraw ()

Constructor.

virtual ~QwtRoundScaleDraw ()

Destructor.

- void setRadius (double radius)
- double radius () const
- void moveCenter (double x, double y)

Move the center of the scale draw, leaving the radius unchanged.

- void moveCenter (const QPointF &)
- · QPointF center () const

Get the center of the scale.

• void setAngleRange (double angle1, double angle2)

Adjust the baseline circle segment for round scales.

· virtual double extent (const QFont &) const

Protected Member Functions

- virtual void drawTick (QPainter *, double val, double len) const
- virtual void drawBackbone (QPainter *) const
- virtual void drawLabel (QPainter *, double val) const

Additional Inherited Members

12.106.1 Detailed Description

A class for drawing round scales.

QwtRoundScaleDraw can be used to draw round scales. The circle segment can be adjusted by setAngleRange(). The geometry of the scale can be specified with moveCenter() and setRadius().

After a scale division has been specified as a QwtScaleDiv object using QwtAbstractScaleDraw::setScaleDiv(const QwtScaleDiv &s), the scale can be drawn with the QwtAbstractScaleDraw::draw() member.

12.106.2 Constructor & Destructor Documentation

12.106.2.1 QwtRoundScaleDraw::QwtRoundScaleDraw ()

Constructor.

The range of the scale is initialized to [0, 100], The center is set to (50, 50) with a radius of 50. The angle range is set to [-135, 135].

12.106.3 Member Function Documentation

12.106.3.1 void QwtRoundScaleDraw::drawBackbone (QPainter * painter) const [protected], [virtual]

Draws the baseline of the scale

Parameters

| painter | Painter |
|---------|---------|

See Also

drawTick(), drawLabel()

Implements QwtAbstractScaleDraw.

```
12.106.3.2 void QwtRoundScaleDraw::drawLabel ( QPainter * painter, double value ) const [protected], [virtual]
```

Draws the label for a major scale tick

Parameters

| painter | Painter |
|---------|---------|
| value | Value |

See Also

drawTick(), drawBackbone()

Implements QwtAbstractScaleDraw.

12.106.3.3 void QwtRoundScaleDraw::drawTick (QPainter * *painter*, **double** *value*, **double** *len*) **const** [protected], [virtual]

Draw a tick

Parameters

| painter | Painter |
|---------|--------------------|
| value | Value of the tick |
| len | Lenght of the tick |

See Also

drawBackbone(), drawLabel()

Implements QwtAbstractScaleDraw.

12.106.3.4 double QwtRoundScaleDraw::extent (const QFont & font) const [virtual]

Calculate the extent of the scale

The extent is the distance between the baseline to the outermost pixel of the scale draw. radius() + extent() is an upper limit for the radius of the bounding circle.

Parameters

| font | Font used for painting the labels |
|------|-----------------------------------|

Returns

Calculated extent

See Also

setMinimumExtent(), minimumExtent()

Warning

The implemented algorithm is not too smart and calculates only an upper limit, that might be a few pixels too large

Implements QwtAbstractScaleDraw.

12.106.3.5 void QwtRoundScaleDraw::moveCenter (const QPointF & center)

Move the center of the scale draw, leaving the radius unchanged

Parameters

| contor | Now contar |
|--------|--------------|
| center | I New Center |
| | |

See Also

setRadius()

12.106.3.6 double QwtRoundScaleDraw::radius () const

Get the radius

Radius is the radius of the backbone without ticks and labels.

Returns

Radius of the scale

See Also

setRadius(), extent()

12.106.3.7 void QwtRoundScaleDraw::setAngleRange (double angle1, double angle2)

Adjust the baseline circle segment for round scales.

The baseline will be drawn from min(angle1,angle2) to max(angle1, angle2). The default setting is [-135, 135]. An angle of 0 degrees corresponds to the 12 o'clock position, and positive angles count in a clockwise direction.

Parameters

| angle1 | |
|--------|--|
| angle2 | boundaries of the angle interval in degrees. |

Warning

- The angle range is limited to [-360, 360] degrees. Angles exceeding this range will be clipped.
- For angles more or equal than 360 degrees above or below min(angle1, angle2), scale marks will not be drawn.
- If you need a counterclockwise scale, use QwtScaleDiv::setInterval()

12.106.3.8 void QwtRoundScaleDraw::setRadius (double radius)

Change of radius the scale

Radius is the radius of the backbone without ticks and labels.

Parameters

```
radius New Radius
```

See Also

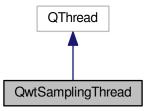
moveCenter()

12.107 QwtSamplingThread Class Reference

A thread collecting samples at regular intervals.

```
#include <qwt_sampling_thread.h>
```

Inheritance diagram for QwtSamplingThread:



Public Slots

- void setInterval (double interval)
- void stop ()

Public Member Functions

- virtual ~QwtSamplingThread ()
 - Destructor.
- double interval () const
- double elapsed () const

Protected Member Functions

- QwtSamplingThread (QObject *parent=NULL)
 - Constructor.
- virtual void run ()
- virtual void sample (double elapsed)=0

```
12.107.1 Detailed Description
```

A thread collecting samples at regular intervals.

Continuous signals are converted into a discrete signal by collecting samples at regular intervals. A discrete signal can be displayed by a QwtPlotSeriesItem on a QwtPlotS

QwtSamplingThread starts a thread calling periodically sample(), to collect and store (or emit) a single sample.

```
See Also
```

```
QwtPlotCurve, QwtPlotSeriesItem
12.107.2 Member Function Documentation
12.107.2.1 double QwtSamplingThread::elapsed ( ) const
Returns
     Time (in ms) since the thread was started
See Also
     QThread::start(), run()
12.107.2.2 double QwtSamplingThread::interval ( ) const
Returns
     Interval (in ms), between 2 calls of sample()
See Also
     setInterval()
12.107.2.3 void QwtSamplingThread::run() [protected], [virtual]
Loop collecting samples started from QThread::start()
See Also
     stop()
12.107.2.4 virtual void QwtSamplingThread::sample (double elapsed) [protected], [pure virtual]
Collect a sample
Parameters
          elapsed
                     Time since the thread was started in milliseconds
12.107.2.5 void QwtSamplingThread::setInterval ( double interval ) [slot]
```

Change the interval (in ms), when sample() is called. The default interval is 1000.0 (= 1s)

Parameters

| interval | Interval |
|----------|----------|
|----------|----------|

See Also

interval()

12.107.2.6 void QwtSamplingThread::stop() [slot]

Terminate the collecting thread

See Also

QThread::start(), run()

12.108 QwtScaleArithmetic Class Reference

Arithmetic including a tolerance.

```
#include <qwt_scale_engine.h>
```

Static Public Member Functions

- static double ceilEps (double value, double intervalSize)
- static double floorEps (double value, double intervalSize)
- static double divideEps (double interval, double steps)

Divide an interval into steps.

• static double divideInterval (double interval, int numSteps, uint base)

12.108.1 Detailed Description

Arithmetic including a tolerance.

12.108.2 Member Function Documentation

12.108.2.1 double QwtScaleArithmetic::ceilEps (double value, double intervalSize) [static]

Ceil a value, relative to an interval

Parameters

| value | Value to be ceiled |
|--------------|--------------------|
| intervalSize | Interval size |

Returns

Rounded value

See Also

floorEps()

12.108.2.2 double QwtScaleArithmetic::divideEps (double intervalSize, double numSteps) [static]

Divide an interval into steps.

 $stepSize = (intervalSize - intervalSize * 10e^{-6})/numSteps$

Parameters

| intervalSize | Interval size |
|--------------|-----------------|
| numSteps | Number of steps |

Returns

Step size

12.108.2.3 double QwtScaleArithmetic::divideInterval (double intervalSize, int numSteps, uint base) [static]

Calculate a step size for a given interval

Parameters

| intervalSize | Interval size |
|--------------|--------------------------------------|
| numSteps | Number of steps |
| base | Base for the division (usually 10) |

Returns

Calculated step size

12.108.2.4 double QwtScaleArithmetic::floorEps (double value, double intervalSize) [static]

Floor a value, relative to an interval

Parameters

| value | Value to be floored |
|--------------|---------------------|
| intervalSize | Interval size |

Returns

Rounded value

See Also

floorEps()

12.109 QwtScaleDiv Class Reference

A class representing a scale division.

```
#include <qwt_scale_div.h>
```

Public Types

enum TickType {
 NoTick = -1, MinorTick, MediumTick, MajorTick,
 NTickTypes }

Scale tick types.

Public Member Functions

- QwtScaleDiv (double lowerBound=0.0, double upperBound=0.0)
- QwtScaleDiv (const QwtInterval &, QList< double >[NTickTypes])

- QwtScaleDiv (double lowerBound, double upperBound, QList< double >[NTickTypes])
- QwtScaleDiv (double lowerBound, double upperBound, const QList< double > &minorTicks, const QList< double > &mediumTicks, const QList< double > &majorTicks)
- bool operator== (const QwtScaleDiv &) const

Equality operator.

• bool operator!= (const QwtScaleDiv &) const

Inequality.

- void setInterval (double lowerBound, double upperBound)
- · void setInterval (const QwtInterval &)
- · QwtInterval interval () const
- void setLowerBound (double)
- double lowerBound () const
- void setUpperBound (double)
- double upperBound () const
- double range () const
- · bool contains (double value) const
- void setTicks (int tickType, const QList< double > &)
- QList< double > ticks (int tickType) const
- bool isEmpty () const

Check if the scale division is empty(lowerBound() == upperBound())

· bool isIncreasing () const

Check if the scale division is increasing(lowerBound() <= upperBound())

- · void invert ()
- · QwtScaleDiv inverted () const
- QwtScaleDiv bounded (double lowerBound, double upperBound) const

12.109.1 Detailed Description

A class representing a scale division.

A Qwt scale is defined by its boundaries and 3 list for the positions of the major, medium and minor ticks.

The upperLimit() might be smaller than the lowerLimit() to indicate inverted scales.

Scale divisions can be calculated from a QwtScaleEngine.

See Also

QwtScaleEngine::divideScale(), QwtPlot::setAxisScaleDiv(), QwtAbstractSlider::setScaleDiv()

12.109.2 Member Enumeration Documentation

12.109.2.1 enum QwtScaleDiv::TickType

Scale tick types.

Enumerator

NoTick No ticks.

MinorTick Minor ticks.

MediumTick Medium ticks.

MajorTick Major ticks.

NTickTypes Number of valid tick types.

12.109.3 Constructor & Destructor Documentation

12.109.3.1 QwtScaleDiv::QwtScaleDiv (double lowerBound = 0 . 0, double upperBound = 0 . 0) [explicit]

Construct a division without ticks

Parameters

| lowerBound | First boundary |
|------------|-----------------|
| upperBound | Second boundary |

Note

lowerBound might be greater than upperBound for inverted scales

12.109.3.2 QwtScaleDiv::QwtScaleDiv (const QwtInterval & interval, QList< double > ticks[NTickTypes]) [explicit]

Construct a scale division

Parameters

| interval | Interval |
|----------|---------------------------------------|
| ticks | List of major, medium and minor ticks |

12.109.3.3 QwtScaleDiv::QwtScaleDiv (double lowerBound, double upperBound, QList< double > ticks[NTickTypes]) [explicit]

Construct a scale division

Parameters

| lowerBound | First boundary |
|------------|---------------------------------------|
| upperBound | Second boundary |
| ticks | List of major, medium and minor ticks |

Note

lowerBound might be greater than upperBound for inverted scales

12.109.3.4 QwtScaleDiv::QwtScaleDiv (double lowerBound, double upperBound, const QList< double > & minorTicks, const QList< double > & majorTicks) [explicit]

Construct a scale division

Parameters

| IowerBound | First boundary |
|-------------|---------------------|
| upperBound | Second boundary |
| minorTicks | List of minor ticks |
| mediumTicks | List medium ticks |
| majorTicks | List of major ticks |

Note

lowerBound might be greater than upperBound for inverted scales

12.109.4 Member Function Documentation

12.109.4.1 QwtScaleDiv QwtScaleDiv::bounded (double lowerBound, double upperBound) const

Return a scale division with an interval [lowerBound, upperBound] where all ticks outside this interval are removed

Parameters

| IowerBound | Lower bound |
|------------|-------------|
| upperBound | Upper bound |

Returns

Scale division with all ticks inside of the given interval

Note

lowerBound might be greater than upperBound for inverted scales

12.109.4.2 bool QwtScaleDiv::contains (double value) const

Return if a value is between lowerBound() and upperBound()

Parameters

| value | Value |
|-------|-------|
| | |

Returns

true/false

12.109.4.3 QwtInterval QwtScaleDiv::interval () const

Returns

lowerBound -> upperBound

12.109.4.4 void QwtScaleDiv::invert ()

Invert the scale division

See Also

inverted()

12.109.4.5 QwtScaleDiv QwtScaleDiv::inverted () const

Returns

A scale division with inverted boundaries and ticks

See Also

invert()

12.109.4.6 double QwtScaleDiv::lowerBound () const

Returns

First boundary

See Also

upperBound()

12.109.4.7 bool QwtScaleDiv::operator!= (const QwtScaleDiv & other) const

Inequality.

Returns

true if this instance is not equal to other

12.109.4.8 bool QwtScaleDiv::operator== (const QwtScaleDiv & other) const

Equality operator.

Returns

true if this instance is equal to other

12.109.4.9 double QwtScaleDiv::range () const

Returns

upperBound() - lowerBound()

12.109.4.10 void QwtScaleDiv::setInterval (double lowerBound, double upperBound)

Change the interval

Parameters

| lowerBound | First boundary |
|------------|-----------------|
| upperBound | Second boundary |

Note

lowerBound might be greater than upperBound for inverted scales

12.109.4.11 void QwtScaleDiv::setInterval (const QwtInterval & interval)

Change the interval

Parameters

interval Interval

12.109.4.12 void QwtScaleDiv::setLowerBound (double lowerBound)

Set the first boundary

Parameters

lowerBound First boundary

See Also

lowerBiound(), setUpperBound()

12.109.4.13 void QwtScaleDiv::setTicks (int $\it type, const QList < double > \& \it ticks$)

Assign ticks

Parameters

| type | MinorTick, MediumTick or MajorTick |
|-------|------------------------------------|
| ticks | Values of the tick positions |

12.109.4.14 void QwtScaleDiv::setUpperBound (double upperBound)

Set the second boundary

Parameters

| upperBound | Second boundary |
|------------|-----------------|
|------------|-----------------|

See Also

upperBound(), setLowerBound()

12.109.4.15 QList < double > QwtScaleDiv::ticks (int type) const

Return a list of ticks

Parameters

| type | MinorTick, MediumTick or MajorTick |
|------|------------------------------------|

Returns

Tick list

12.109.4.16 double QwtScaleDiv::upperBound () const

Returns

upper bound

See Also

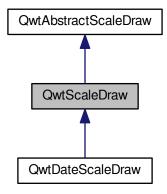
lowerBound()

12.110 QwtScaleDraw Class Reference

A class for drawing scales.

#include <qwt_scale_draw.h>

Inheritance diagram for QwtScaleDraw:



Public Types

• enum Alignment { BottomScale, TopScale, LeftScale, RightScale }

Public Member Functions

· QwtScaleDraw ()

Constructor.

• virtual \sim QwtScaleDraw ()

Destructor.

· void getBorderDistHint (const QFont &, int &start, int &end) const

Determine the minimum border distance.

- int minLabelDist (const QFont &) const
- · int minLength (const QFont &) const
- · virtual double extent (const QFont &) const
- void move (double x, double y)
- void move (const QPointF &)

Move the position of the scale.

- void setLength (double length)
- · Alignment alignment () const
- void setAlignment (Alignment)
- · Qt::Orientation orientation () const
- · QPointF pos () const
- double length () const
- void setLabelAlignment (Qt::Alignment)

Change the label flags.

- Qt::Alignment labelAlignment () const
- void setLabelRotation (double rotation)
- · double labelRotation () const
- int maxLabelHeight (const QFont &) const
- int maxLabelWidth (const QFont &) const
- QPointF labelPosition (double val) const
- QRectF labelRect (const QFont &, double val) const

- · QSizeF labelSize (const QFont &, double val) const
- QRect boundingLabelRect (const QFont &, double val) const

Find the bounding rectangle for the label.

Protected Member Functions

- QTransform labelTransformation (const QPointF &, const QSizeF &) const
- virtual void drawTick (QPainter *, double val, double len) const
- virtual void drawBackbone (QPainter *) const
- virtual void drawLabel (QPainter *, double val) const

12.110.1 Detailed Description

A class for drawing scales.

QwtScaleDraw can be used to draw linear or logarithmic scales. A scale has a position, an alignment and a length, which can be specified. The labels can be rotated and aligned to the ticks using setLabelRotation() and setLabelAlignment().

After a scale division has been specified as a QwtScaleDiv object using QwtAbstractScaleDraw::setScaleDiv(const QwtScaleDiv &s), the scale can be drawn with the QwtAbstractScaleDraw::draw() member.

```
12.110.2 Member Enumeration Documentation
```

12.110.2.1 enum QwtScaleDraw::Alignment

Alignment of the scale draw

See Also

setAlignment(), alignment()

Enumerator

BottomScale The scale is below.

TopScale The scale is above.

LeftScale The scale is left.

RightScale The scale is right.

12.110.3 Constructor & Destructor Documentation

12.110.3.1 QwtScaleDraw::QwtScaleDraw()

Constructor.

The range of the scale is initialized to [0, 100], The position is at (0, 0) with a length of 100. The orientation is QwtAbstractScaleDraw::Bottom.

12.110.4 Member Function Documentation

12.110.4.1 QwtScaleDraw::Alignment QwtScaleDraw::alignment () const

Return alignment of the scale

See Also

setAlignment()

Returns

Alignment of the scale

12.110.4.2 QRect QwtScaleDraw::boundingLabelRect (const QFont & font, double value) const

Find the bounding rectangle for the label.

The coordinates of the rectangle are absolute (calculated from pos()). in direction of the tick.

Parameters

| font | Font used for painting |
|-------|------------------------|
| value | Value |

Returns

Bounding rectangle

See Also

labelRect()

12.110.4.3 void QwtScaleDraw::drawBackbone(QPainter * painter) const [protected], [virtual]

Draws the baseline of the scale

Parameters

| painter | Painter |
|---------|---------|

See Also

drawTick(), drawLabel()

Implements QwtAbstractScaleDraw.

12.110.4.4 void QwtScaleDraw::drawLabel (QPainter * painter, double value) const [protected], [virtual]

Draws the label for a major scale tick

Parameters

| painter | Painter |
|---------|---------|
| value | Value |

See Also

drawTick(), drawBackbone(), boundingLabelRect()

Implements QwtAbstractScaleDraw.

12.110.4.5 void QwtScaleDraw::drawTick (QPainter * *painter*, double *value*, double *len*) const [protected], [virtual]

Draw a tick

Parameters

| painter | Painter |
|---------|--------------------|
| value | Value of the tick |
| len | Length of the tick |

See Also

drawBackbone(), drawLabel()

Implements QwtAbstractScaleDraw.

12.110.4.6 double QwtScaleDraw::extent (const QFont & font) const [virtual]

Calculate the width/height that is needed for a vertical/horizontal scale.

The extent is calculated from the pen width of the backbone, the major tick length, the spacing and the maximum width/height of the labels.

Parameters

| font | Font used for painting the labels |
|------|-----------------------------------|

Returns

Extent

See Also

minLength()

Implements QwtAbstractScaleDraw.

12.110.4.7 void QwtScaleDraw::getBorderDistHint (const QFont & font, int & start, int & end) const

Determine the minimum border distance.

This member function returns the minimum space needed to draw the mark labels at the scale's endpoints.

Parameters

| font | Font |
|-------|-----------------------|
| start | Start border distance |
| end | End border distance |

12.110.4.8 Qt::Alignment QwtScaleDraw::labelAlignment () const

Returns

the label flags

See Also

setLabelAlignment(), labelRotation()

12.110.4.9 QPointF QwtScaleDraw::labelPosition (double value) const

Find the position, where to paint a label

The position has a distance that depends on the length of the ticks in direction of the alignment().

Parameters

| value | Value |
|-------|-------|
|-------|-------|

Returns

Position, where to paint a label

12.110.4.10 QRectF QwtScaleDraw::labelRect (const QFont & font, double value) const

Find the bounding rectangle for the label. The coordinates of the rectangle are relative to spacing + tick length from the backbone in direction of the tick.

Parameters

| font | Font used for painting |
|-------|------------------------|
| value | Value |

Returns

Bounding rectangle that is needed to draw a label

12.110.4.11 double QwtScaleDraw::labelRotation () const

Returns

the label rotation

See Also

setLabelRotation(), labelAlignment()

12.110.4.12 QSizeF QwtScaleDraw::labelSize (const QFont & font, double value) const

Calculate the size that is needed to draw a label

Parameters

| font | Label font |
|-------|------------|
| value | Value |

Returns

Size that is needed to draw a label

12.110.4.13 QTransform QwtScaleDraw::labelTransformation (const QPointF & pos, const QSizeF & size) const [protected]

Calculate the transformation that is needed to paint a label depending on its alignment and rotation.

Parameters

| pos | Position where to paint the label |
|------|-----------------------------------|
| size | Size of the label |

Returns

Transformation matrix

See Also

setLabelAlignment(), setLabelRotation()

12.110.4.14 double QwtScaleDraw::length () const

Returns

the length of the backbone

See Also

setLength(), pos()

12.110.4.15 int QwtScaleDraw::maxLabelHeight (const QFont & font) const

Parameters

font | Font

Returns

the maximum height of a label

12.110.4.16 int QwtScaleDraw::maxLabelWidth (const QFont & font) const

Parameters

font | Font

Returns

the maximum width of a label

12.110.4.17 int QwtScaleDraw::minLabelDist (const QFont & font) const

Determine the minimum distance between two labels, that is necessary that the texts don't overlap.

Parameters

font Font

Returns

The maximum width of a label

See Also

getBorderDistHint()

12.110.4.18 int QwtScaleDraw::minLength (const QFont & font) const

Calculate the minimum length that is needed to draw the scale

Parameters

font Font used for painting the labels

Returns

Minimum length that is needed to draw the scale

See Also

extent()

12.110.4.19 void QwtScaleDraw::move (double x, double y) [inline]

Move the position of the scale

Parameters

| X | X coordinate |
|---|--------------|
| у | Y coordinate |

See Also

```
move(const QPointF &)
```

12.110.4.20 void QwtScaleDraw::move (const QPointF & pos)

Move the position of the scale.

The meaning of the parameter pos depends on the alignment:

QwtScaleDraw::LeftScale The origin is the topmost point of the backbone. The backbone is a vertical line. Scale marks and labels are drawn at the left of the backbone.

QwtScaleDraw::RightScale The origin is the topmost point of the backbone. The backbone is a vertical line. Scale marks and labels are drawn at the right of the backbone.

QwtScaleDraw::TopScale The origin is the leftmost point of the backbone. The backbone is a horizontal line. Scale marks and labels are drawn above the backbone.

QwtScaleDraw::BottomScale The origin is the leftmost point of the backbone. The backbone is a horizontal line Scale marks and labels are drawn below the backbone.

Parameters

| pos | Origin of the scale |
|-----|---------------------|
|-----|---------------------|

See Also

```
pos(), setLength()
```

12.110.4.21 Qt::Orientation QwtScaleDraw::orientation () const

Return the orientation

TopScale, BottomScale are horizontal (Qt::Horizontal) scales, LeftScale, RightScale are vertical (Qt::Vertical) scales.

Returns

Orientation of the scale

See Also

alignment()

12.110.4.22 QPointF QwtScaleDraw::pos () const

Returns

Origin of the scale

See Also

move(), length()

12.110.4.23 void QwtScaleDraw::setAlignment (Alignment align)

Set the alignment of the scale

| align | Alignment of the scale |
|-------|------------------------|
|-------|------------------------|

The default alignment is QwtScaleDraw::BottomScale

See Also

alignment()

12.110.4.24 void QwtScaleDraw::setLabelAlignment (Qt::Alignment alignment)

Change the label flags.

Labels are aligned to the point tick length + spacing away from the backbone.

The alignment is relative to the orientation of the label text. In case of an flags of 0 the label will be aligned depending on the orientation of the scale:

```
QwtScaleDraw::TopScale: Qt::AlignHCenter | Qt::AlignTop\n
QwtScaleDraw::BottomScale: Qt::AlignHCenter | Qt::AlignBottom\n
QwtScaleDraw::LeftScale: Qt::AlignLeft | Qt::AlignVCenter\n
QwtScaleDraw::RightScale: Qt::AlignRight | Qt::AlignVCenter\n
```

Changing the alignment is often necessary for rotated labels.

Parameters

| alignment | Or'd Qt::AlignmentFlags see <qnamespace.h></qnamespace.h> |
|-----------|---|

See Also

setLabelRotation(), labelRotation(), labelAlignment()

Warning

The various alignments might be confusing. The alignment of the label is not the alignment of the scale and is not the alignment of the flags (QwtText::flags()) returned from QwtAbstractScaleDraw::label().

12.110.4.25 void QwtScaleDraw::setLabelRotation (double rotation)

Rotate all labels.

When changing the rotation, it might be necessary to adjust the label flags too. Finding a useful combination is often the result of try and error.

Parameters

| rotation | Angle in degrees. When changing the label rotation, the label flags often needs to be adjusted |
|----------|--|
| | too. |

See Also

setLabelAlignment(), labelRotation(), labelAlignment().

12.110.4.26 void QwtScaleDraw::setLength (double length)

Set the length of the backbone.

The length doesn't include the space needed for overlapping labels.

Parameters

| length | Length of the backbone |
|--------|------------------------|
|--------|------------------------|

See Also

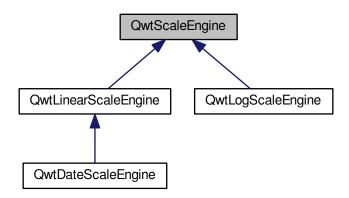
move(), minLabelDist()

12.111 QwtScaleEngine Class Reference

Base class for scale engines.

```
#include <qwt_scale_engine.h>
```

Inheritance diagram for QwtScaleEngine:



Public Types

- enum Attribute {
 NoAttribute = 0x00, IncludeReference = 0x01, Symmetric = 0x02, Floating = 0x04, Inverted = 0x08 }
- typedef QFlags < Attribute > Attributes
 Layout attributes.

Public Member Functions

- QwtScaleEngine (uint base=10)
- virtual ∼QwtScaleEngine ()

Destructor.

- void setBase (uint base)
- uint base () const
- void setAttribute (Attribute, bool on=true)
- bool testAttribute (Attribute) const
- · void setAttributes (Attributes)
- · Attributes attributes () const
- void setReference (double reference)

Specify a reference point.

- double reference () const
- void setMargins (double lower, double upper)

Specify margins at the scale's endpoints.

- · double lowerMargin () const
- double upperMargin () const
- virtual void autoScale (int maxNumSteps, double &x1, double &x2, double &stepSize) const =0
- virtual QwtScaleDiv divideScale (double x1, double x2, int maxMajorSteps, int maxMinorSteps, double step-Size=0.0) const =0

Calculate a scale division.

- void setTransformation (QwtTransform *)
- QwtTransform * transformation () const

Protected Member Functions

- bool contains (const QwtInterval &, double val) const
- QList< double > strip (const QList< double > &, const QwtInterval &) const
- double divideInterval (double interval, int numSteps) const
- · QwtInterval buildInterval (double v) const

Build an interval around a value.

12.111.1 Detailed Description

Base class for scale engines.

A scale engine tries to find "reasonable" ranges and step sizes for scales.

The layout of the scale can be varied with setAttribute().

Qwt offers implementations for logarithmic and linear scales.

12.111.2 Member Enumeration Documentation

12.111.2.1 enum QwtScaleEngine::Attribute

Layout attributes

See Also

setAttribute(), testAttribute(), reference(), lowerMargin(), upperMargin()

Enumerator

NoAttribute No attributes.

IncludeReference Build a scale which includes the reference() value.

Symmetric Build a scale which is symmetric to the reference() value.

Floating The endpoints of the scale are supposed to be equal the outmost included values plus the specified margins (see setMargins()). If this attribute is *not* set, the endpoints of the scale will be integer multiples of the step size.

Inverted Turn the scale upside down.

12.111.3 Constructor & Destructor Documentation

12.111.3.1 QwtScaleEngine::QwtScaleEngine (uint base = 10) [explicit]

Constructor

Parameters

| base | Base of the scale engine |
|------|--------------------------|

See Also

setBase()

12.111.4 Member Function Documentation

12.111.4.1 QwtScaleEngine::Attributes QwtScaleEngine::attributes () const

Returns

Scale attributes

See Also

Attribute, setAttributes(), testAttribute()

12.111.4.2 virtual void QwtScaleEngine::autoScale (int maxNumSteps, double & x1, double & x2, double & stepSize) const [pure virtual]

Align and divide an interval

Parameters

| maxNumSteps | Max. number of steps |
|-------------|---------------------------------------|
| x1 | First limit of the interval (In/Out) |
| x2 | Second limit of the interval (In/Out) |
| stepSize | Step size (Return value) |

Implemented in QwtLogScaleEngine, QwtLinearScaleEngine, and QwtDateScaleEngine.

12.111.4.3 uint QwtScaleEngine::base () const

Returns

base Base of the scale engine

See Also

setBase()

12.111.4.4 QwtInterval QwtScaleEngine::buildInterval (double value) const [protected]

Build an interval around a value.

In case of v == 0.0 the interval is [-0.5, 0.5], otherwide it is [0.5 * v, 1.5 * v]

Parameters

| value Initial value | |
|---------------------|--|
|---------------------|--|

Returns

Calculated interval

12.111.4.5 bool QwtScaleEngine::contains (const QwtInterval & interval, double value) const [protected]

Check if an interval "contains" a value

| interval | Interval |
|----------|----------|
| value | Value |

Returns

True, when the value is inside the interval

12.111.4.6 double QwtScaleEngine::divideInterval (double intervalSize, int numSteps) const [protected]

Calculate a step size for an interval size

Parameters

| intervalSize | Interval size |
|--------------|-----------------|
| numSteps | Number of steps |

Returns

Step size

12.111.4.7 virtual QwtScaleDiv QwtScaleEngine::divideScale (double x1, double x2, int maxMajorSteps, int maxMinorSteps, double stepSize = 0.0) const [pure virtual]

Calculate a scale division.

Parameters

| x1 | First interval limit |
|---------------|--|
| x2 | Second interval limit |
| maxMajorSteps | Maximum for the number of major steps |
| maxMinorSteps | Maximum number of minor steps |
| stepSize | Step size. If stepSize == 0.0, the scaleEngine calculates one. |

Returns

Calculated scale division

Implemented in QwtLogScaleEngine, QwtLinearScaleEngine, and QwtDateScaleEngine.

12.111.4.8 double QwtScaleEngine::lowerMargin () const

Returns

the margin at the lower end of the scale The default margin is 0.

See Also

setMargins()

12.111.4.9 double QwtScaleEngine::reference () const

Returns

the reference value

See Also

setReference(), setAttribute()

12.111.4.10 void QwtScaleEngine::setAttribute (Attribute attribute, bool on = true)

Change a scale attribute

Parameters

| attribute | Attribute to change |
|-----------|---------------------|
| on | On/Off |

See Also

Attribute, testAttribute()

12.111.4.11 void QwtScaleEngine::setAttributes (Attributes attributes)

Change the scale attribute

Parameters

| attributes | Set scale attributes |
|------------|----------------------|

See Also

Attribute, attributes()

12.111.4.12 void QwtScaleEngine::setBase (uint base)

Set the base of the scale engine

While a base of 10 is what 99.9% of all applications need certain scales might need a different base: f.e 2

The default setting is 10

Parameters

| base | Base of the engine |
|------|--------------------|

See Also

base()

12.111.4.13 void QwtScaleEngine::setMargins (double lower, double upper)

Specify margins at the scale's endpoints.

Parameters

| lower | minimum distance between the scale's lower boundary and the smallest enclosed value |
|-------|---|
| upper | minimum distance between the scale's upper boundary and the greatest enclosed value |

Margins can be used to leave a minimum amount of space between the enclosed intervals and the boundaries of the scale.

Warning

QwtLogScaleEngine measures the margins in decades.

See Also

upperMargin(), lowerMargin()

12.111.4.14 void QwtScaleEngine::setReference (double r)

Specify a reference point.

r new reference value

The reference point is needed if options IncludeReference or Symmetric are active. Its default value is 0.0.

See Also

Attribute

12.111.4.15 void QwtScaleEngine::setTransformation (QwtTransform * transform)

Assign a transformation

Parameters

| transform | Transformation |
|-----------|----------------|
|-----------|----------------|

The transformation object is used as factory for clones that are returned by transformation()

The scale engine takes ownership of the transformation.

See Also

QwtTransform::copy(), transformation()

12.111.4.16 QList< double > QwtScaleEngine::strip (const QList< double > & ticks, const QwtInterval & interval) const [protected]

Remove ticks from a list, that are not inside an interval

Parameters

| ticks | Tick list |
|----------|-----------|
| interval | Interval |

Returns

Stripped tick list

12.111.4.17 bool QwtScaleEngine::testAttribute (Attribute attribute) const

Returns

True, if attribute is enabled.

Parameters

| attribute | Attribute to be tested |
|-----------|------------------------|

See Also

Attribute, setAttribute()

12.111.4.18 QwtTransform * QwtScaleEngine::transformation () const

Create and return a clone of the transformation of the engine. When the engine has no special transformation NULL is returned, indicating no transformation.

Returns

A clone of the transfomation

See Also

setTransformation()

```
12.111.4.19 double QwtScaleEngine::upperMargin ( ) const
```

Returns

the margin at the upper end of the scale The default margin is 0.

See Also

setMargins()

12.112 QwtScaleMap Class Reference

A scale map.

```
#include <qwt_scale_map.h>
```

Public Member Functions

· QwtScaleMap ()

Constructor.

QwtScaleMap (const QwtScaleMap &)

Copy constructor.

- ∼QwtScaleMap ()
- QwtScaleMap & operator= (const QwtScaleMap &)

Assignment operator.

- void setTransformation (QwtTransform *)
- const QwtTransform * transformation () const

Get the transformation.

• void setPaintInterval (double p1, double p2)

Specify the borders of the paint device interval.

void setScaleInterval (double s1, double s2)

Specify the borders of the scale interval.

- double transform (double s) const
- double invTransform (double p) const
- double p1 () const
- double p2 () const
- double s1 () const
- double s2 () const
- double pDist () const
- double sDist () const
- bool isInverting () const

Static Public Member Functions

- static QRectF transform (const QwtScaleMap &, const QwtScaleMap &, const QRectF &)
- static QRectF invTransform (const QwtScaleMap &, const QwtScaleMap &, const QRectF &)
- static QPointF transform (const QwtScaleMap &, const QwtScaleMap &, const QPointF &)
- static QPointF invTransform (const QwtScaleMap &, const QwtScaleMap &, const QPointF &)

12.112.1 Detailed Description

A scale map.

QwtScaleMap offers transformations from the coordinate system of a scale into the linear coordinate system of a paint device and vice versa.

12.112.2 Constructor & Destructor Documentation

12.112.2.1 QwtScaleMap::QwtScaleMap()

Constructor.

The scale and paint device intervals are both set to [0,1].

12.112.2.2 QwtScaleMap::~QwtScaleMap()

Destructor

12.112.3 Member Function Documentation

12.112.3.1 double QwtScaleMap::invTransform (double p) const [inline]

Transform an paint device value into a value in the interval of the scale.

Parameters

| p Value relative to the coordinates of the paint device | |
|---|--|
|---|--|

Returns

Transformed value

See Also

transform()

12.112.3.2 QRectF QwtScaleMap::invTransform (const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & rect) [static]

Transform a rectangle from paint to scale coordinates

Parameters

| | хМар | X map |
|---|------|--------------------------------|
| | уМар | Y map |
| ſ | rect | Rectangle in paint coordinates |

Returns

Rectangle in scale coordinates

See Also

transform()

12.112.3.3 QPointF QwtScaleMap::invTransform (const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QPointF & pos) [static]

Transform a rectangle from paint to scale coordinates

Parameters

| хМар | X map |
|------|-------------------------------|
| уМар | Y map |
| pos | Position in paint coordinates |

```
Returns
     Position in scale coordinates
See Also
     transform()
12.112.3.4 bool QwtScaleMap::isInverting ( ) const [inline]
Returns
     True, when (p1() < p2()) != (s1() < s2())
12.112.3.5 double QwtScaleMap::p1 ( ) const [inline]
Returns
     First border of the paint interval
12.112.3.6 double QwtScaleMap::p2( ) const [inline]
Returns
     Second border of the paint interval
12.112.3.7 double QwtScaleMap::pDist() const [inline]
Returns
     qwtAbs(p2() - p1())
12.112.3.8 double QwtScaleMap::s1() const [inline]
Returns
     First border of the scale interval
12.112.3.9 double QwtScaleMap::s2( ) const [inline]
Returns
     Second border of the scale interval
12.112.3.10 double QwtScaleMap::sDist() const [inline]
Returns
     qwtAbs(s2() - s1())
12.112.3.11 void QwtScaleMap::setPaintInterval ( double p1, double p2 )
Specify the borders of the paint device interval.
```

| p1 | first border |
|----|---------------|
| p2 | second border |

12.112.3.12 void QwtScaleMap::setScaleInterval (double s1, double s2)

Specify the borders of the scale interval.

Parameters

| s1 | first border |
|----|---------------|
| s2 | second border |

Warning

scales might be aligned to transformation depending boundaries

12.112.3.13 void QwtScaleMap::setTransformation (QwtTransform * transform)

Initialize the map with a transformation

12.112.3.14 double QwtScaleMap::transform (double s) const [inline]

Transform a point related to the scale interval into an point related to the interval of the paint device

Parameters

| s | Value relative to the coordinates of the scale |
|---|--|
|---|--|

Returns

Transformed value

See Also

invTransform()

12.112.3.15 QRectF QwtScaleMap::transform (const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRectF & rect) [static]

Transform a rectangle from scale to paint coordinates

Parameters

| хМар | X map |
|------|--------------------------------|
| уМар | Y map |
| rect | Rectangle in scale coordinates |

Returns

Rectangle in paint coordinates

See Also

invTransform()

12.112.3.16 QPointF QwtScaleMap::transform (const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QPointF & pos) [static]

Transform a point from scale to paint coordinates

Parameters

| хМар | X map |
|------|-------------------------------|
| уМар | Y map |
| pos | Position in scale coordinates |

Returns

Position in paint coordinates

See Also

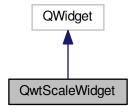
invTransform()

12.113 QwtScaleWidget Class Reference

A Widget which contains a scale.

#include <qwt_scale_widget.h>

Inheritance diagram for QwtScaleWidget:



Public Types

• enum LayoutFlag { TitleInverted = 1 }

Layout flags of the title.

typedef QFlags < LayoutFlags

Layout flags of the title.

Signals

• void scaleDivChanged ()

Signal emitted, whenever the scale division changes.

Public Member Functions

QwtScaleWidget (QWidget *parent=NULL)

Create a scale with the position QwtScaleWidget::Left.

• QwtScaleWidget (QwtScaleDraw::Alignment, QWidget *parent=NULL)

Constructor.

virtual ~QwtScaleWidget ()

Destructor.

- void setTitle (const QString &title)
- void setTitle (const QwtText &title)
- QwtText title () const
- void setLayoutFlag (LayoutFlag, bool on)
- · bool testLayoutFlag (LayoutFlag) const
- void setBorderDist (int start, int end)
- · int startBorderDist () const
- int endBorderDist () const
- void getBorderDistHint (int &start, int &end) const

Calculate a hint for the border distances.

- · void getMinBorderDist (int &start, int &end) const
- void setMinBorderDist (int start, int end)
- void setMargin (int)

Specify the margin to the colorBar/base line.

- int margin () const
- void setSpacing (int td)

Specify the distance between color bar, scale and title.

- int spacing () const
- void setScaleDiv (const QwtScaleDiv &sd)

Assign a scale division.

- void setTransformation (QwtTransform *)
- void setScaleDraw (QwtScaleDraw *)
- const QwtScaleDraw * scaleDraw () const
- QwtScaleDraw * scaleDraw ()
- void setLabelAlignment (Qt::Alignment)

Change the alignment for the labels.

• void setLabelRotation (double rotation)

Change the rotation for the labels. See QwtScaleDraw::setLabelRotation().

- void setColorBarEnabled (bool)
- · bool isColorBarEnabled () const
- void setColorBarWidth (int)
- int colorBarWidth () const
- void setColorMap (const QwtInterval &, QwtColorMap *)
- · QwtInterval colorBarInterval () const
- const QwtColorMap * colorMap () const
- virtual QSize sizeHint () const
- virtual QSize minimumSizeHint () const
- int titleHeightForWidth (int width) const

Find the height of the title for a given width.

• int dimForLength (int length, const QFont &scaleFont) const

Find the minimum dimension for a given length. dim is the height, length the width seen in direction of the title.

- void drawColorBar (QPainter *painter, const QRectF &) const
- void drawTitle (QPainter *painter, QwtScaleDraw::Alignment, const QRectF &rect) const
- void setAlignment (QwtScaleDraw::Alignment)
- QwtScaleDraw::Alignment alignment () const
- QRectF colorBarRect (const QRectF &) const

Protected Member Functions

virtual void paintEvent (QPaintEvent *)

paintEvent

- virtual void resizeEvent (QResizeEvent *)
- void draw (QPainter *p) const

draw the scale

· void scaleChange ()

Notify a change of the scale.

void layoutScale (bool update=true)

12.113.1 Detailed Description

A Widget which contains a scale.

This Widget can be used to decorate composite widgets with a scale.

12.113.2 Member Enumeration Documentation

12.113.2.1 enum QwtScaleWidget::LayoutFlag

Layout flags of the title.

Enumerator

TitleInverted The title of vertical scales is painted from top to bottom. Otherwise it is painted from bottom to top.

12.113.3 Constructor & Destructor Documentation

12.113.3.1 QwtScaleWidget::QwtScaleWidget (QWidget * parent = NULL) [explicit]

Create a scale with the position QwtScaleWidget::Left.

Parameters

| parent | Parent widget |
|--------|---------------|

12.113.3.2 QwtScaleWidget::QwtScaleWidget (QwtScaleDraw::Alignment align, QWidget * parent = NULL) [explicit]

Constructor.

Parameters

| align | Alignment. |
|--------|---------------|
| parent | Parent widget |

12.113.4 Member Function Documentation

12.113.4.1 QwtScaleDraw::Alignment QwtScaleWidget::alignment () const

Returns

position

See Also

setPosition()

12.113.4.2 QwtInterval QwtScaleWidget::colorBarInterval () const

Returns

Value interval for the color bar

See Also

setColorMap(), colorMap()

12.113.4.3 QRectF QwtScaleWidget::colorBarRect (const QRectF & rect) const

Calculate the the rectangle for the color bar

Parameters

| rect Bounding rectangle for all components of the scale | |
|---|--|
|---|--|

Returns

Rectangle for the color bar

12.113.4.4 int QwtScaleWidget::colorBarWidth () const

Returns

Width of the color bar

See Also

setColorBarEnabled(), setColorBarEnabled()

12.113.4.5 const QwtColorMap * QwtScaleWidget::colorMap () const

Returns

Color map

See Also

setColorMap(), colorBarInterval()

12.113.4.6 int QwtScaleWidget::dimForLength (int length, const QFont & scaleFont) const

Find the minimum dimension for a given length. dim is the height, length the width seen in direction of the title.

Parameters

| length | width for horizontal, height for vertical scales |
|-----------|--|
| scaleFont | Font of the scale |

Returns

height for horizontal, width for vertical scales

12.113.4.7 void QwtScaleWidget::drawColorBar (QPainter * painter, const QRectF & rect) const

Draw the color bar of the scale widget

Parameters

| painter | Painter |
|---------|--------------------------------------|
| rect | Bounding rectangle for the color bar |

See Also

setColorBarEnabled()

12.113.4.8 void QwtScaleWidget::drawTitle (QPainter * painter, QwtScaleDraw::Alignment align, const QRectF & rect) const

Rotate and paint a title according to its position into a given rectangle.

Parameters

| painter | Painter |
|---------|--------------------|
| align | Alignment |
| rect | Bounding rectangle |

12.113.4.9 int QwtScaleWidget::endBorderDist () const

Returns

end border distance

See Also

setBorderDist()

12.113.4.10 void QwtScaleWidget::getBorderDistHint (int & start, int & end) const

Calculate a hint for the border distances.

This member function calculates the distance of the scale's endpoints from the widget borders which is required for the mark labels to fit into the widget. The maximum of this distance an the minimum border distance is returned.

Parameters

| start | Return parameter for the border width at the beginning of the scale |
|-------|---|
| end | Return parameter for the border width at the end of the scale |

Warning

• The minimum border distance depends on the font.

See Also

setMinBorderDist(), getMinBorderDist(), setBorderDist()

12.113.4.11 void QwtScaleWidget::getMinBorderDist (int & start, int & end) const

Get the minimum value for the distances of the scale's endpoints from the widget borders.

Parameters

| start | Return parameter for the border width at the beginning of the scale |
|-------|---|
| end | Return parameter for the border width at the end of the scale |

See Also

setMinBorderDist(), getBorderDistHint()

12.113.4.12 bool QwtScaleWidget::isColorBarEnabled () const

Returns

true, when the color bar is enabled

See Also

setColorBarEnabled(), setColorBarWidth()

12.113.4.13 void QwtScaleWidget::layoutScale (bool update_geometry = true) [protected]

Recalculate the scale's geometry and layout based on the current geometry and fonts.

Parameters

| update | Notify the layout system and call update to redraw the scale |
|----------|--|
| geometry | |

12.113.4.14 int QwtScaleWidget::margin () const

Returns

margin

See Also

setMargin()

12.113.4.15 QSize QwtScaleWidget::minimumSizeHint() const [virtual]

Returns

a minimum size hint

12.113.4.16 void QwtScaleWidget::resizeEvent (QResizeEvent * event) [protected], [virtual]

Event handler for resize events

Parameters

event Resize event

12.113.4.17 void QwtScaleWidget::scaleChange() [protected]

Notify a change of the scale.

This virtual function can be overloaded by derived classes. The default implementation updates the geometry and repaints the widget.

12.113.4.18 const QwtScaleDraw * QwtScaleWidget::scaleDraw () const

Returns

scaleDraw of this scale

See Also

setScaleDraw(), QwtScaleDraw::setScaleDraw()

12.113.4.19 QwtScaleDraw * QwtScaleWidget::scaleDraw ()

Returns

scaleDraw of this scale

See Also

QwtScaleDraw::setScaleDraw()

12.113.4.20 void QwtScaleWidget::setAlignment (QwtScaleDraw::Alignment alignment)

Change the alignment

Parameters

| alignment | New alignment |
|-----------|---------------|
|-----------|---------------|

See Also

alignment()

12.113.4.21 void QwtScaleWidget::setBorderDist (int dist1, int dist2)

Specify distances of the scale's endpoints from the widget's borders. The actual borders will never be less than minimum border distance.

Parameters

| dist1 | Left or top Distance |
|-------|--------------------------|
| dist2 | Right or bottom distance |

See Also

borderDist()

12.113.4.22 void QwtScaleWidget::setColorBarEnabled (bool on)

En/disable a color bar associated to the scale

See Also

isColorBarEnabled(), setColorBarWidth()

12.113.4.23 void QwtScaleWidget::setColorBarWidth (int width)

Set the width of the color bar

| width | Width |
|-------|-------|
|-------|-------|

See Also

colorBarWidth(), setColorBarEnabled()

12.113.4.24 void QwtScaleWidget::setColorMap (const QwtInterval & interval, QwtColorMap * colorMap)

Set the color map and value interval, that are used for displaying the color bar.

Parameters

| interval | Value interval |
|----------|----------------|
| colorMap | Color map |

See Also

colorMap(), colorBarInterval()

12.113.4.25 void QwtScaleWidget::setLabelAlignment (Qt::Alignment alignment)

Change the alignment for the labels.

See Also

QwtScaleDraw::setLabelAlignment(), setLabelRotation()

12.113.4.26 void QwtScaleWidget::setLabelRotation (double rotation)

Change the rotation for the labels. See QwtScaleDraw::setLabelRotation().

Parameters

| rotation | Rotation |
|----------|----------|

See Also

QwtScaleDraw::setLabelRotation(), setLabelFlags()

12.113.4.27 void QwtScaleWidget::setLayoutFlag (LayoutFlag flag, bool on)

Toggle an layout flag

Parameters

| flag | Layout flag |
|------|-------------|
| on | true/false |

See Also

testLayoutFlag(), LayoutFlag

12.113.4.28 void QwtScaleWidget::setMargin (int margin)

Specify the margin to the colorBar/base line.

Parameters

| margin | Margin |
|--------|--------|

See Also

margin()

12.113.4.29 void QwtScaleWidget::setMinBorderDist (int start, int end)

Set a minimum value for the distances of the scale's endpoints from the widget borders. This is useful to avoid that the scales are "jumping", when the tick labels or their positions change often.

Parameters

| start | Minimum for the start border |
|-------|------------------------------|
| end | Minimum for the end border |

See Also

getMinBorderDist(), getBorderDistHint()

12.113.4.30 void QwtScaleWidget::setScaleDiv (const QwtScaleDiv & scaleDiv)

Assign a scale division.

The scale division determines where to set the tick marks.

Parameters

| scaleDiv | Scale Division |
|----------|----------------|
|----------|----------------|

See Also

For more information about scale divisions, see QwtScaleDiv.

12.113.4.31 void QwtScaleWidget::setScaleDraw (QwtScaleDraw * scaleDraw)

Set a scale draw

scaleDraw has to be created with new and will be deleted in \sim QwtScaleWidget() or the next call of setScaleDraw(). scaleDraw will be initialized with the attributes of the previous scaleDraw object.

Parameters

| scaleDraw | ScaleDraw object |
|-----------|------------------|

See Also

scaleDraw()

12.113.4.32 void QwtScaleWidget::setSpacing (int spacing)

Specify the distance between color bar, scale and title.

Parameters

spacing Spacing

See Also

spacing()

12.113.4.33 void QwtScaleWidget::setTitle (const QString & title)

Give title new text contents

Parameters

title New title

See Also

title(), setTitle(const QwtText &);

12.113.4.34 void QwtScaleWidget::setTitle (const QwtText & title)

Give title new text contents

Parameters

title New title

See Also

title()

Warning

The title flags are interpreted in direction of the label, AlignTop, AlignBottom can't be set as the title will always be aligned to the scale.

12.113.4.35 void QwtScaleWidget::setTransformation (QwtTransform * transformation)

Set the transformation

Parameters

transformation Transformation

See Also

QwtAbstractScaleDraw::scaleDraw(), QwtScaleMap

12.113.4.36 QSize QwtScaleWidget::sizeHint() const [virtual]

Returns

a size hint

12.113.4.37 int QwtScaleWidget::spacing () const

Returns

distance between scale and title

See Also

setMargin()

12.113.4.38 int QwtScaleWidget::startBorderDist () const

Returns

start border distance

See Also

setBorderDist()

12.113.4.39 bool QwtScaleWidget::testLayoutFlag (LayoutFlag flag) const

Test a layout flag

Parameters

flag Layout flag

Returns

true/false

See Also

setLayoutFlag(), LayoutFlag

12.113.4.40 QwtText QwtScaleWidget::title () const

Returns

title

See Also

setTitle()

12.113.4.41 int QwtScaleWidget::titleHeightForWidth (int width) const

Find the height of the title for a given width.

Parameters

width Width

Returns

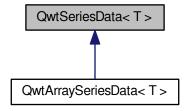
height Height

12.114 QwtSeriesData < T > Class Template Reference

Abstract interface for iterating over samples.

#include <qwt_series_data.h>

Inheritance diagram for QwtSeriesData< T >:



Public Member Functions

QwtSeriesData ()

Constructor.

virtual ~QwtSeriesData ()

Destructor.

- virtual size_t size () const =0
- virtual T sample (size_t i) const =0
- virtual QRectF boundingRect () const =0
- virtual void setRectOfInterest (const QRectF &rect)

Protected Attributes

· QRectF d boundingRect

Can be used to cache a calculated bounding rectangle.

12.114.1 Detailed Description

template<typename T>class QwtSeriesData< T>

Abstract interface for iterating over samples.

Qwt offers several implementations of the QwtSeriesData API, but in situations, where data of an application specific format needs to be displayed, without having to copy it, it is recommended to implement an individual data access.

A subclass of QwtSeriesData<QPointF> must implement:

• size()

Should return number of data points.

• sample()

Should return values x and y values of the sample at specific position as QPointF object.

• boundingRect()

Should return the bounding rectangle of the data series. It is used for autoscaling and might help certain algorithms for displaying the data. You can use qwtBoundingRect() for an implementation but often it is possible to implement a more efficient algorithm depending on the characteristics of the series. The member d_boundingRect is intended for caching the calculated rectangle.

12.114.2 Member Function Documentation

12.114.2.1 template < typename T > virtual QRectF QwtSeriesData < T >::boundingRect() const [pure virtual]

Calculate the bounding rect of all samples

The bounding rect is necessary for autoscaling and can be used for a couple of painting optimizations.

qwtBoundingRect(...) offers slow implementations iterating over the samples. For large sets it is recommended to implement something faster f.e. by caching the bounding rectangle.

Returns

Bounding rectangle

Implemented in QwtTradingChartData, QwtSetSeriesData, QwtIntervalSeriesData, QwtPoint3DSeriesData, QwtPointSeriesData, QwtCPointerData, and QwtPointArrayData.

12.114.2.2 template < typename T > virtual T QwtSeriesData < T > ::sample (size_t i) const [pure virtual]

Return a sample

Parameters

| ; | Index |
|---|-------|
| I | Index |
| | |

Returns

Sample at position i

Implemented in QwtArraySeriesData< T >, QwtArraySeriesData< QwtIntervalSample >, QwtArraySeriesData< QwtOHLCSample >, QwtArraySeriesData< QPointF >, QwtArraySeriesData< QwtPoint3D >, QwtArraySeriesData< QwtSetSample >, QwtSyntheticPointData, QwtCPointerData, and QwtPointArrayData.

12.114.2.3 template < typename T > void QwtSeriesData < T >::setRectOfInterest (const QRectF & rect) [virtual]

Set a the "rect of interest"

QwtPlotSeriesItem defines the current area of the plot canvas as "rectangle of interest" (QwtPlotSeriesItem::update-ScaleDiv()). It can be used to implement different levels of details.

The default implementation does nothing.

Parameters

rect Rectangle of interest

Reimplemented in QwtSyntheticPointData.

12.114.2.4 template<typename T> virtual size_t QwtSeriesData<T>::size() const [pure virtual]

Returns

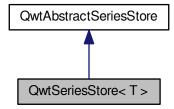
Number of samples

12.115 QwtSeriesStore < T > Class Template Reference

Class storing a QwtSeriesData object.

#include <qwt_series_store.h>

Inheritance diagram for QwtSeriesStore< T >:



Public Member Functions

· QwtSeriesStore ()

Constructor The store contains no series.

∼QwtSeriesStore ()

Destructor.

- void setData (QwtSeriesData< T > *series)
- QwtSeriesData< T > * data ()
- const QwtSeriesData< T > * data () const
- T sample (int index) const
- virtual size_t dataSize () const
- virtual QRectF dataRect () const
- virtual void setRectOfInterest (const QRectF &rect)
- QwtSeriesData< T > * swapData (QwtSeriesData< T > *series)

Additional Inherited Members

12.115.1 Detailed Description

template<typename T>class QwtSeriesStore< T>

Class storing a QwtSeriesData object.

QwtSeriesStore and QwtPlotSeriesItem are intended as base classes for all plot items iterating over a series of samples. Both classes share a virtual base class (QwtAbstractSeriesStore) to bridge between them.

QwtSeriesStore offers the template based part for the plot item API, so that QwtPlotSeriesItem can be derived without any hassle with templates.

12.115.2 Member Function Documentation

12.115.2.1 template < typename T > QwtSeriesData < T > * QwtSeriesStore < T >::data() [inline]

Returns

the the series data

 $\textbf{12.115.2.2} \quad \textbf{template} < \textbf{typename} \ \textbf{T} > \textbf{const} \ \textbf{QwtSeriesData} < \textbf{T} > * \ \textbf{QwtSeriesStore} < \textbf{T} > :: \textbf{data()} \ \textbf{const} \quad \texttt{[inline]}$

Returns

the the series data

12.115.2.3 template < typename T > QRectF QwtSeriesStore < T >::dataRect() const [virtual]

Returns

Bounding rectangle of the series or an invalid rectangle, when no series is stored

See Also

QwtSeriesData<T>::boundingRect()

Implements QwtAbstractSeriesStore.

12.115.2.4 template < typename T > size_t QwtSeriesStore < T >::dataSize() const [virtual]

Returns

Number of samples of the series

See Also

setData(), QwtSeriesData<T>::size()

Implements QwtAbstractSeriesStore.

12.115.2.5 template<typename T > T QwtSeriesStore< T >::sample (int index) const [inline]

Parameters

| index | Index |
|-------|-------|

Returns

Sample at position index

 $12.115.2.6 \quad template < typename \ T > void \ QwtSeriesStore < T > ::setData (\ QwtSeriesData < T > * \textit{series} \)$

Assign a series of samples

Parameters

| series | Data |
|--------|------|
|--------|------|

Warning

The item takes ownership of the data object, deleting it when its not used anymore.

12.115.2.7 template < typename T > void QwtSeriesStore < T >::setRectOfInterest (const QRectF & rect) [virtual]

Set a the "rect of interest" for the series

rect Rectangle of interest

See Also

QwtSeriesData<T>::setRectOfInterest()

Implements QwtAbstractSeriesStore.

```
12.115.2.8 template<typename T> QwtSeriesData< T> * QwtSeriesStore< T>::swapData ( QwtSeriesData< T > * series )
```

Replace a series without deleting the previous one

Parameters

series New series

Returns

Previously assigned series

12.116 QwtSetSample Class Reference

A sample of the types (x1...xn, y) or (x, y1..yn)

```
#include <qwt_samples.h>
```

Public Member Functions

- QwtSetSample ()
- QwtSetSample (double, const QVector< double > &=QVector< double >())
- bool operator== (const QwtSetSample &other) const

Compare operator.

• bool operator!= (const QwtSetSample &other) const

Compare operator.

· double added () const

Public Attributes

double value

value

QVector< double > set

Vector of values associated to value.

12.116.1 Detailed Description

A sample of the types (x1...xn, y) or (x, y1..yn)

12.116.2 Constructor & Destructor Documentation

12.116.2.1 QwtSetSample::QwtSetSample() [inline]

Constructor The value is set to 0.0

12.116.2.2 QwtSetSample::QwtSetSample (double v, const QVector < double > & s = QVector < double > ()) [inline]

Constructor

| V | Value |
|---|---------------|
| s | Set of values |

12.116.3 Member Function Documentation

12.116.3.1 double QwtSetSample::added() const [inline]

Returns

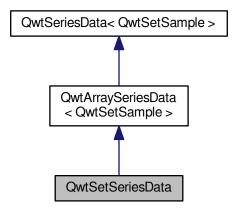
All values of the set added

12.117 QwtSetSeriesData Class Reference

Interface for iterating over an array of samples.

#include <qwt_series_data.h>

Inheritance diagram for QwtSetSeriesData:



Public Member Functions

- QwtSetSeriesData (const QVector< QwtSetSample > &=QVector< QwtSetSample >())
- virtual QRectF boundingRect () const

Calculate the bounding rectangle.

Additional Inherited Members

12.117.1 Detailed Description

Interface for iterating over an array of samples.

12.117.2 Constructor & Destructor Documentation

12.117.2.1 QwtSetSeriesData::QwtSetSeriesData (const QVector< QwtSetSample > & samples = QVector<QwtSetSample>())

Constructor

| samples Samp | bles |
|--------------|------|
|--------------|------|

12.117.3 Member Function Documentation

12.117.3.1 QRectF QwtSetSeriesData::boundingRect () const [virtual]

Calculate the bounding rectangle.

The bounding rectangle is calculated once by iterating over all points and is stored for all following requests.

Returns

Bounding rectangle

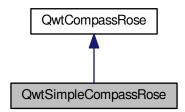
Implements QwtSeriesData < QwtSetSample >.

12.118 QwtSimpleCompassRose Class Reference

A simple rose for QwtCompass.

#include <qwt_compass_rose.h>

Inheritance diagram for QwtSimpleCompassRose:



Public Member Functions

- QwtSimpleCompassRose (int numThorns=8, int numThornLevels=-1)
- virtual ~QwtSimpleCompassRose ()

Destructor.

- void setWidth (double w)
- double width () const
- void setNumThorns (int count)
- int numThorns () const
- void setNumThornLevels (int count)
- int numThornLevels () const
- void setShrinkFactor (double factor)
- double shrinkFactor () const
- virtual void draw (QPainter *, const QPointF ¢er, double radius, double north, QPalette::ColorGroup=Q-Palette::Active) const

Static Public Member Functions

• static void drawRose (QPainter *, const QPalette &, const QPointF ¢er, double radius, double origin, double width, int numThorns, int numThornLevels, double shrinkFactor)

12.118.1 Detailed Description

A simple rose for QwtCompass.

12.118.2 Constructor & Destructor Documentation

12.118.2.1 QwtSimpleCompassRose::QwtSimpleCompassRose (int numThorns = 8, int numThornLevels = -1)

Constructor

Parameters

| numThorns | Number of thorns |
|----------------|------------------------|
| numThornLevels | Number of thorn levels |

12.118.3 Member Function Documentation

12.118.3.1 void QwtSimpleCompassRose::draw (QPainter * painter, const QPointF & center, double radius, double north,

QPalette::ColorGroup cg = QPalette::Active) const [virtual]

Draw the rose

Parameters

| painter | Painter |
|---------|--------------------|
| center | Center point |
| radius | Radius of the rose |
| north | Position |
| cg | Color group |

Implements QwtCompassRose.

12.118.3.2 void QwtSimpleCompassRose::drawRose (QPainter * painter, const QPalette & palette, const QPointF & center, double radius, double north, double width, int numThorns, int numThornLevels, double shrinkFactor)
[static]

Draw the rose

Parameters

| painter | Painter |
|----------------|---|
| palette | Palette |
| center | Center of the rose |
| radius | Radius of the rose |
| north | Position pointing to north |
| width | Width of the rose |
| numThorns | Number of thorns |
| numThornLevels | Number of thorn levels |
| shrinkFactor | Factor to shrink the thorns with each level |

12.118.3.3 int QwtSimpleCompassRose::numThornLevels () const

Returns

Number of thorn levels

See Also

setNumThorns(), setNumThornLevels()

12.118.3.4 int QwtSimpleCompassRose::numThorns () const

Returns

Number of thorns

See Also

setNumThorns(), setNumThornLevels()

12.118.3.5 void QwtSimpleCompassRose::setNumThornLevels (int numThornLevels)

Set the of thorns levels

Parameters

numThornLevels Number of thorns levels

See Also

setNumThorns(), numThornLevels()

12.118.3.6 void QwtSimpleCompassRose::setNumThorns (int numThorns)

Set the number of thorns on one level The number is aligned to a multiple of 4, with a minimum of 4

Parameters

numThorns Number of thorns

See Also

numThorns(), setNumThornLevels()

12.118.3.7 void QwtSimpleCompassRose::setShrinkFactor (double factor)

Set the Factor how to shrink the thorns with each level The default value is 0.9.

Parameters

factor | Shrink factor

See Also

shrinkFactor()

12.118.3.8 void QwtSimpleCompassRose::setWidth (double width)

Set the width of the rose heads. Lower value make thinner heads. The range is limited from 0.03 to 0.4.

Parameters

| width | Width |
|-------|-------|

12.118.3.9 double QwtSimpleCompassRose::shrinkFactor () const

Returns

Factor how to shrink the thorns with each level

See Also

setShrinkFactor()

12.118.3.10 double QwtSimpleCompassRose::width () const

Returns

Width of the rose

See Also

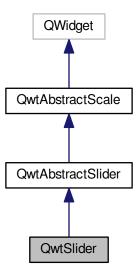
setWidth()

12.119 QwtSlider Class Reference

The Slider Widget.

#include <qwt_slider.h>

Inheritance diagram for QwtSlider:



Public Types

• enum ScalePosition { NoScale, LeadingScale, TrailingScale }

Public Member Functions

- QwtSlider (QWidget *parent=NULL)
- QwtSlider (Qt::Orientation, QWidget *parent=NULL)
- virtual ~QwtSlider ()

Destructor.

void setOrientation (Qt::Orientation)

Set the orientation.

- · Qt::Orientation orientation () const
- void setScalePosition (ScalePosition)

Change the position of the scale.

- · ScalePosition scalePosition () const
- void setTrough (bool)
- bool hasTrough () const
- void setGroove (bool)
- bool hasGroove () const
- void setHandleSize (const QSize &)

Set the slider's handle size.

- · QSize handleSize () const
- void setBorderWidth (int bw)

Change the slider's border width.

- int borderWidth () const
- void setSpacing (int)

Change the spacing between trough and scale.

- int spacing () const
- · virtual QSize sizeHint () const
- virtual QSize minimumSizeHint () const
- void setScaleDraw (QwtScaleDraw *)

Set a scale draw.

- const QwtScaleDraw * scaleDraw () const
- void setUpdateInterval (int)

Specify the update interval for automatic scrolling.

• int updateInterval () const

Protected Member Functions

• virtual double scrolledTo (const QPoint &) const

Determine the value for a new position of the slider handle.

virtual bool isScrollPosition (const QPoint &) const

Determine what to do when the user presses a mouse button.

- virtual void drawSlider (QPainter *, const QRect &) const
- virtual void drawHandle (QPainter *, const QRect &, int pos) const
- virtual void mousePressEvent (QMouseEvent *)
- virtual void mouseReleaseEvent (QMouseEvent *)
- virtual void resizeEvent (QResizeEvent *)
- virtual void paintEvent (QPaintEvent *)
- virtual void changeEvent (QEvent *)
- virtual void timerEvent (QTimerEvent *)
- virtual void scaleChange ()

Notify changed scale.

- · QRect sliderRect () const
- · QRect handleRect () const

Additional Inherited Members

12.119.1 Detailed Description

The Slider Widget.

QwtSlider is a slider widget which operates on an interval of type double. Its position is related to a scale showing the current value.

The slider can be customized by having a through, a groove - or both.

12.119.2 Member Enumeration Documentation

12.119.2.1 enum QwtSlider::ScalePosition

Position of the scale

See Also

QwtSlider(), setScalePosition(), setOrientation()

Enumerator

NoScale The slider has no scale.

LeadingScale The scale is right of a vertical or below a horizontal slider.

TrailingScale The scale is left of a vertical or above a horizontal slider.

12.119.3 Constructor & Destructor Documentation

12.119.3.1 QwtSlider::QwtSlider(QWidget*parent=NULL) [explicit]

Construct vertical slider in QwtSlider::Trough style with a scale to the left.

The scale is initialized to [0.0, 100.0] and the value set to 0.0.

Parameters

| parent | Parent widget |
|--------|---------------|

See Also

setOrientation(), setScalePosition(), setBackgroundStyle()

12.119.3.2 QwtSlider::QwtSlider (Qt::Orientation orientation, QWidget * parent = NULL) [explicit]

Construct a slider in QwtSlider::Trough style

When orientation is Qt::Vertical the scale will be aligned to the left - otherwise at the top of the slider.

The scale is initialized to [0.0, 100.0] and the value set to 0.0.

Parameters

| parent | Parent widget |
|-------------|----------------------------|
| orientation | Orientation of the slider. |

12.119.4 Member Function Documentation

12.119.4.1 int QwtSlider::borderWidth () const

Returns

the border width.

See Also

setBorderWidth()

12.119.4.2 void QwtSlider::changeEvent (QEvent * event) [protected], [virtual]

Handles QEvent::StyleChange and QEvent::FontChange events

Parameters

| event | Change event |
|-------|--------------|

12.119.4.3 void QwtSlider::drawHandle (QPainter * painter, const QRect & handleRect, int pos) const [protected], [virtual]

Draw the thumb at a position

Parameters

| painter | Painter |
|------------|---|
| handleRect | Bounding rectangle of the handle |
| pos | Position of the handle marker in widget coordinates |

12.119.4.4 void QwtSlider::drawSlider (QPainter * *painter***, const QRect &** *sliderRect* **) const** [protected], [virtual]

Draw the slider into the specified rectangle.

Parameters

| painter | Painter |
|------------|----------------------------------|
| sliderRect | Bounding rectangle of the slider |

12.119.4.5 QRect QwtSlider::handleRect() const [protected]

Returns

Bounding rectangle of the slider handle

12.119.4.6 QSize QwtSlider::handleSize () const

Returns

Size of the handle.

See Also

setHandleSize()

12.119.4.7 bool QwtSlider::hasGroove () const

Returns

True, when the groove is visisble

See Also

setGroove(), hasTrough()

12.119.4.8 bool QwtSlider::hasTrough () const

Returns

True, when the trough is visisble

See Also

setTrough(), hasGroove()

12.119.4.9 bool QwtSlider::isScrollPosition (const QPoint & pos) const [protected], [virtual]

Determine what to do when the user presses a mouse button.

Parameters

pos Mouse position

Return values

True, when handle Rect() contains pos

See Also

scrolledTo()

Implements QwtAbstractSlider.

12.119.4.10 QSize QwtSlider::minimumSizeHint() const [virtual]

Returns

Minimum size hint

See Also

sizeHint()

12.119.4.11 void QwtSlider::mousePressEvent (QMouseEvent * event) [protected], [virtual]

Mouse press event handler

Parameters

event | Mouse event

Reimplemented from QwtAbstractSlider.

12.119.4.12 void QwtSlider::mouseReleaseEvent (QMouseEvent * event) [protected], [virtual]

Mouse release event handler

Parameters

event Mouse event

Reimplemented from QwtAbstractSlider.

12.119.4.13 Qt::Orientation QwtSlider::orientation () const

Returns

Orientation

```
See Also
```

setOrientation()

12.119.4.14 void QwtSlider::paintEvent (QPaintEvent * event) [protected], [virtual]

Qt paint event handler

Parameters

event Paint event

12.119.4.15 void QwtSlider::resizeEvent (QResizeEvent * event) [protected], [virtual]

Qt resize event handler

Parameters

event Resize event

12.119.4.16 const QwtScaleDraw * QwtSlider::scaleDraw () const

Returns

the scale draw of the slider

See Also

setScaleDraw()

12.119.4.17 QwtSlider::ScalePosition QwtSlider::scalePosition () const

Returns

Position of the scale

See Also

setScalePosition()

12.119.4.18 double QwtSlider::scrolledTo(const QPoint & pos) const [protected], [virtual]

Determine the value for a new position of the slider handle.

Parameters

pos Mouse position

Returns

Value for the mouse position

See Also

isScrollPosition()

Implements QwtAbstractSlider.

12.119.4.19 void QwtSlider::setBorderWidth (int width)

Change the slider's border width.

The border width is used for drawing the slider handle and the trough.

Parameters

width Border width

See Also

borderWidth()

12.119.4.20 void QwtSlider::setGroove (bool on)

En/Disable the groove

The slider can be cutomized by showing a groove for the handle.

Parameters

on When true, the groove is visible

See Also

hasGroove(), setThrough()

12.119.4.21 void QwtSlider::setHandleSize (const QSize & size)

Set the slider's handle size.

When the size is empty the slider handle will be painted with a default size depending on its orientation() and backgroundStyle().

Parameters

size New size

See Also

handleSize()

12.119.4.22 void QwtSlider::setOrientation (Qt::Orientation orientation)

Set the orientation.

Parameters

orientation Allowed values are Qt::Horizontal and Qt::Vertical.

See Also

orientation(), scalePosition()

12.119.4.23 void QwtSlider::setScaleDraw (QwtScaleDraw * scaleDraw)

Set a scale draw.

For changing the labels of the scales, it is necessary to derive from QwtScaleDraw and overload QwtScaleDraw::label().

Parameters

scaleDraw ScaleDraw object, that has to be created with new and will be deleted in ~QwtSlider() or the next call of setScaleDraw().

See Also

scaleDraw()

12.119.4.24 void QwtSlider::setScalePosition (ScalePosition scalePosition)

Change the position of the scale.

Parameters

scalePosition | Position of the scale.

See Also

ScalePosition, scalePosition()

12.119.4.25 void QwtSlider::setSpacing (int spacing)

Change the spacing between trough and scale.

A spacing of 0 means, that the backbone of the scale is covered by the trough.

The default setting is 4 pixels.

Parameters

spacing Number of pixels

See Also

spacing();

12.119.4.26 void QwtSlider::setTrough (bool on)

En/Disable the trough

The slider can be cutomized by showing a trough for the handle.

Parameters

on When true, the groove is visible

See Also

hasTrough(), setGroove()

12.119.4.27 void QwtSlider::setUpdateInterval (int interval)

Specify the update interval for automatic scrolling.

The minimal accepted value is 50 ms.

Parameters

```
Update interval in milliseconds
           interval
See Also
     setUpdateInterval()
12.119.4.28 QSize QwtSlider::sizeHint() const [virtual]
Returns
     minimumSizeHint()
12.119.4.29 QRect QwtSlider::sliderRect() const [protected]
Returns
     Bounding rectangle of the slider - without the scale
12.119.4.30 int QwtSlider::spacing ( ) const
Returns
     Number of pixels between slider and scale
See Also
     setSpacing()
12.119.4.31 void QwtSlider::timerEvent ( QTimerEvent * event ) [protected], [virtual]
Timer event handler
Handles the timer, when the mouse stays pressed inside the sliderRect().
Parameters
                     Mouse event
             event
12.119.4.32 int QwtSlider::updateInterval ( ) const
Returns
     Update interval in milliseconds for automatic scrolling
See Also
     setUpdateInterval()
12.120
        QwtSpline Class Reference
A class for spline interpolation.
#include <qwt_spline.h>
Public Types

    enum SplineType { Natural, Periodic }

         Spline type.
```

Public Member Functions

QwtSpline ()

Constructor.

- QwtSpline (const QwtSpline &)
- ∼QwtSpline ()

Destructor.

- QwtSpline & operator= (const QwtSpline &)
- void setSplineType (SplineType)
- SplineType splineType () const
- bool setPoints (const QPolygonF &points)

Calculate the spline coefficients.

- QPolygonF points () const
- · void reset ()

Free allocated memory and set size to 0.

• bool isValid () const

True if valid.

- double value (double x) const
- const QVector< double > & coefficientsA () const
- const QVector< double > & coefficientsB () const
- const QVector< double > & coefficientsC () const

Protected Member Functions

bool buildNaturalSpline (const QPolygonF &)

Determines the coefficients for a natural spline.

bool buildPeriodicSpline (const QPolygonF &)

Determines the coefficients for a periodic spline.

12.120.1 Detailed Description

A class for spline interpolation.

The QwtSpline class is used for cubical spline interpolation. Two types of splines, natural and periodic, are supported.

Usage:

- 1. First call setPoints() to determine the spline coefficients for a tabulated function y(x).
- 2. After the coefficients have been set up, the interpolated function value for an argument x can be determined by calling QwtSpline::value().

Example:

```
#include <qwt_spline.h>

QPolygonF interpolate(const QPolygonF& points, int numValues)
{
    QwtSpline spline;
    if ( !spline.setPoints(points) )
        return points;

    QPolygonF interpolatedPoints(numValues);

    const double delta =
        (points[numPoints - 1].x() - points[0].x()) / (points.size() - 1);
    for(i = 0; i < points.size(); i++) / interpolate
    {
        const double x = points[0].x() + i * delta;
        interpolatedPoints[i].setX(x);
        interpolatedPoints[i].setY(spline.value(x));
    }
    return interpolatedPoints;
}</pre>
```

12.120.2 Member Enumeration Documentation 12.120.2.1 enum QwtSpline::SplineType Spline type. Enumerator Natural A natural spline. Periodic A periodic spline. 12.120.3 Constructor & Destructor Documentation 12.120.3.1 QwtSpline::QwtSpline (const QwtSpline & other) Copy constructor **Parameters** other | Spline used for initialization 12.120.4 Member Function Documentation 12.120.4.1 bool QwtSpline::buildNaturalSpline (const QPolygonF & points) [protected] Determines the coefficients for a natural spline. Returns true if successful 12.120.4.2 bool QwtSpline::buildPeriodicSpline (const QPolygonF & points) [protected] Determines the coefficients for a periodic spline. Returns true if successful 12.120.4.3 const QVector < double > & QwtSpline::coefficientsA () const Returns A coefficients 12.120.4.4 const QVector < double > & QwtSpline::coefficientsB () const Returns B coefficients 12.120.4.5 const QVector < double > & QwtSpline::coefficientsC () const Returns C coefficients 12.120.4.6 QwtSpline & QwtSpline::operator= (const QwtSpline & other)

Assignment operator

Parameters

other | Spline used for initialization

Returns

*this

12.120.4.7 QPolygonF QwtSpline::points () const

Returns

Points, that have been by setPoints()

12.120.4.8 bool QwtSpline::setPoints (const QPolygonF & points)

Calculate the spline coefficients.

Depending on the value of *periodic*, this function will determine the coefficients for a natural or a periodic spline and store them internally.

Parameters

points Points

Returns

true if successful

Warning

The sequence of x (but not y) values has to be strictly monotone increasing, which means points[i].x() < points[i+1].x(). If this is not the case, the function will return false

12.120.4.9 void QwtSpline::setSplineType (SplineType splineType)

Select the algorithm used for calculating the spline

Parameters

splineType | Spline type

See Also

splineType()

12.120.4.10 QwtSpline::SplineType QwtSpline::splineType () const

Returns

the spline type

See Also

setSplineType()

12.120.4.11 double QwtSpline::value (double x) const

Calculate the interpolated function value corresponding to a given argument x.

Parameters

x Coordinate

Returns

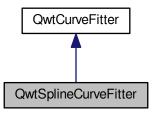
Interpolated coordinate

12.121 QwtSplineCurveFitter Class Reference

A curve fitter using cubic splines.

```
#include <qwt_curve_fitter.h>
```

Inheritance diagram for QwtSplineCurveFitter:



Public Types

• enum FitMode { Auto, Spline, ParametricSpline }

Public Member Functions

• QwtSplineCurveFitter ()

Constructor.

virtual ~QwtSplineCurveFitter ()

Destructor.

- void setFitMode (FitMode)
- FitMode fitMode () const
- void setSpline (const QwtSpline &)
- const QwtSpline & spline () const
- QwtSpline & spline ()
- void setSplineSize (int size)
- int splineSize () const
- virtual QPolygonF fitCurve (const QPolygonF &) const

Additional Inherited Members

12.121.1 Detailed Description

A curve fitter using cubic splines.

12.121.2 Member Enumeration Documentation

12.121.2.1 enum QwtSplineCurveFitter::FitMode

Spline type The default setting is Auto

See Also

setFitMode(), FitMode()

Enumerator

Auto Use the default spline algorithm for polygons with increasing x values (p[i-1] < p[i]), otherwise use a parametric spline algorithm.

Spline Use a default spline algorithm.

ParametricSpline Use a parametric spline algorithm.

12.121.3 Member Function Documentation

12.121.3.1 QPolygonF QwtSplineCurveFitter::fitCurve (const QPolygonF & points) const [virtual]

Find a curve which has the best fit to a series of data points

Parameters

points Series of data points

Returns

Curve points

Implements QwtCurveFitter.

12.121.3.2 QwtSplineCurveFitter::FitMode QwtSplineCurveFitter::fitMode () const

Returns

Mode representing a spline algorithm

See Also

setFitMode()

12.121.3.3 void QwtSplineCurveFitter::setFitMode (FitMode mode)

Select the algorithm used for building the spline

Parameters

mode | Mode representing a spline algorithm

See Also

fitMode()

12.121.3.4 void QwtSplineCurveFitter::setSpline (const QwtSpline & spline)

Assign a spline

```
Parameters
```

```
spline Spline
```

See Also

spline()

12.121.3.5 void QwtSplineCurveFitter::setSplineSize (int splineSize)

Assign a spline size (has to be at least 10 points)

Parameters

```
splineSize | Spline size
```

See Also

splineSize()

12.121.3.6 const QwtSpline & QwtSplineCurveFitter::spline () const

Returns

Spline

See Also

setSpline()

12.121.3.7 QwtSpline & QwtSplineCurveFitter::spline ()

Returns

Spline

See Also

setSpline()

12.121.3.8 int QwtSplineCurveFitter::splineSize () const

Returns

Spline size

See Also

setSplineSize()

12.122 QwtSymbol Class Reference

A class for drawing symbols.

#include <qwt_symbol.h>

Public Types

```
    enum Style {
        NoSymbol = -1, Ellipse, Rect, Diamond,
        Triangle, DTriangle, UTriangle, LTriangle,
        RTriangle, Cross, XCross, HLine,
        VLine, Star1, Star2, Hexagon,
        Path, Pixmap, Graphic, SvgDocument,
        UserStyle = 1000 }
    enum CachePolicy { NoCache, Cache, AutoCache }
```

Public Member Functions

- QwtSymbol (Style=NoSymbol)
- QwtSymbol (Style, const QBrush &, const QPen &, const QSize &)

Constructor.

QwtSymbol (const QPainterPath &, const QBrush &, const QPen &)

Constructor.

virtual ~QwtSymbol ()

Destructor.

- void setCachePolicy (CachePolicy)
- · CachePolicy cachePolicy () const
- void setSize (const QSize &)
- void setSize (int width, int height=-1)

Specify the symbol's size.

- · const QSize & size () const
- void setPinPoint (const QPointF &pos, bool enable=true)

Set and enable a pin point.

- QPointF pinPoint () const
- void setPinPointEnabled (bool)
- bool isPinPointEnabled () const
- virtual void setColor (const QColor &)

Set the color of the symbol.

void setBrush (const QBrush &b)

Assign a brush.

- const QBrush & brush () const
- void setPen (const QColor &, greal width=0.0, Qt::PenStyle=Qt::SolidLine)
- void setPen (const QPen &)
- const QPen & pen () const
- void setStyle (Style)
- Style style () const
- void setPath (const QPainterPath &)

Set a painter path as symbol.

- const QPainterPath & path () const
- void setPixmap (const QPixmap &)
- const QPixmap & pixmap () const
- void setGraphic (const QwtGraphic &)
- · const QwtGraphic & graphic () const
- void setSvgDocument (const QByteArray &)
- void drawSymbol (QPainter *, const QRectF &) const

Draw the symbol into a rectangle.

void drawSymbol (QPainter *, const QPointF &) const

Draw the symbol at a specified position.

void drawSymbols (QPainter *, const QPolygonF &) const

Draw symbols at the specified points.

- void drawSymbols (QPainter *, const QPointF *, int numPoints) const
- · virtual QRect boundingRect () const
- void invalidateCache ()

Protected Member Functions

virtual void renderSymbols (QPainter *, const QPointF *, int numPoints) const

12.122.1 Detailed Description

A class for drawing symbols.

12.122.2 Member Enumeration Documentation

12.122.2.1 enum QwtSymbol::CachePolicy

Depending on the render engine and the complexity of the symbol shape it might be faster to render the symbol to a pixmap and to paint this pixmap.

F.e. the raster paint engine is a pure software renderer where in cache mode a draw operation usually ends in raster operation with the backing store, that are usually faster, than the algorithms for rendering polygons. But the opposite can be expected for graphic pipelines that can make use of hardware acceleration.

The default setting is AutoCache

See Also

setCachePolicy(), cachePolicy()

Note

The policy has no effect, when the symbol is painted to a vector graphics format (PDF, SVG).

Warning

Since Qt 4.8 raster is the default backend on X11

Enumerator

NoCache Don't use a pixmap cache.

Cache Always use a pixmap cache.

AutoCache Use a cache when one of the following conditions is true:

• The symbol is rendered with the software renderer (QPaintEngine::Raster)

12.122.2.2 enum QwtSymbol::Style

Symbol Style

```
See Also
```

```
setStyle(), style()
```

Enumerator

NoSymbol No Style. The symbol cannot be drawn.

Ellipse or circle.

Rect Rectangle.

Diamond Diamond.

Triangle Triangle pointing upwards.

DTriangle Triangle pointing downwards.

UTriangle Triangle pointing upwards.

LTriangle Triangle pointing left.

RTriangle Triangle pointing right.

Cross (+)

XCross Diagonal cross (X)

HLine Horizontal line.

VLine Vertical line.

Star1 X combined with +.

Star2 Six-pointed star.

Hexagon Hexagon.

Path The symbol is represented by a painter path, where the origin (0, 0) of the path coordinate system is mapped to the position of the symbol.

See Also

setPath(), path()

Pixmap The symbol is represented by a pixmap. The pixmap is centered or aligned to its pin point.

See Also

setPinPoint()

Graphic The symbol is represented by a graphic. The graphic is centered or aligned to its pin point.

See Also

setPinPoint()

SvgDocument The symbol is represented by a SVG graphic. The graphic is centered or aligned to its pin point.

See Also

setPinPoint()

UserStyle Styles >= QwtSymbol::UserSymbol are reserved for derived classes of QwtSymbol that overload drawSymbols() with additional application specific symbol types.

12.122.3 Constructor & Destructor Documentation

12.122.3.1 QwtSymbol::QwtSymbol (Style style = NoSymbol)

Default Constructor

Parameters

| style | Symbol Style |
|-------|--------------|

The symbol is constructed with gray interior, black outline with zero width, no size and style 'NoSymbol'.

12.122.3.2 QwtSymbol::QwtSymbol::Style style, const QBrush & brush, const QPen & pen, const QSize & size)

Constructor.

Parameters

| style | Symbol Style |
|-------|----------------------------|
| brush | brush to fill the interior |
| pen | outline pen |
| size | size |

See Also

```
setStyle(), setBrush(), setPen(), setSize()
```

12.122.3.3 QwtSymbol::QwtSymbol (const QPainterPath & path, const QBrush & brush, const QPen & pen)

Constructor.

The symbol gets initialized by a painter path. The style is set to QwtSymbol::Path, the size is set to empty (the path is displayed unscaled).

Parameters

| path | painter path |
|-------|----------------------------|
| brush | brush to fill the interior |
| pen | outline pen |

See Also

```
setPath(), setBrush(), setPen(), setSize()
```

12.122.4 Member Function Documentation

12.122.4.1 QRect QwtSymbol::boundingRect () const [virtual]

Calculate the bounding rectangle for a symbol at position (0,0).

Returns

Bounding rectangle

12.122.4.2 const QBrush & QwtSymbol::brush () const

Returns

Brush

See Also

setBrush()

12.122.4.3 QwtSymbol::CachePolicy QwtSymbol::cachePolicy () const

Returns

Cache policy

See Also

CachePolicy, setCachePolicy()

12.122.4.4 void QwtSymbol::drawSymbol (QPainter * painter, const QRectF & rect) const

Draw the symbol into a rectangle.

The symbol is painted centered and scaled into the target rectangle. It is always painted uncached and the pin point is ignored.

This method is primarily intended for drawing a symbol to the legend.

Parameters

| painter | Painter |
|---------|---------------------------------|
| rect | Target rectangle for the symbol |

12.122.4.5 void QwtSymbol::drawSymbol (QPainter * painter, const QPointF & pos) const [inline]

Draw the symbol at a specified position.

Parameters

| painter | Painter |
|---------|--|
| pos | Position of the symbol in screen coordinates |

12.122.4.6 void QwtSymbol::drawSymbols (QPainter * painter, const QPolygonF & points) const [inline]

Draw symbols at the specified points.

Parameters

| painter | Painter |
|---------|--|
| points | Positions of the symbols in screen coordinates |

12.122.4.7 void QwtSymbol::drawSymbols (QPainter * painter, const QPointF * points, int numPoints) const

Render an array of symbols

Painting several symbols is more effective than drawing symbols one by one, as a couple of layout calculations and setting of pen/brush can be done once for the complete array.

Parameters

| painte | Painter |
|-----------|------------------|
| points | Array of points |
| numPoints | Number of points |

12.122.4.8 const QwtGraphic & QwtSymbol::graphic () const

Returns

Assigned graphic

```
See Also
      setGraphic()
12.122.4.9 void QwtSymbol::invalidateCache ( )
Invalidate the cached symbol pixmap
The symbol invalidates its cache, whenever an attribute is changed that has an effect ob how to display a symbol. In
case of derived classes with individual styles ( >= QwtSymbol::UserStyle ) it might be necessary to call invalidate-
Cache() for attributes that are relevant for this style.
See Also
      CachePolicy, setCachePolicy(), drawSymbols()
12.122.4.10 bool QwtSymbol::isPinPointEnabled ( ) const
Returns
      True, when the pin point translation is enabled
See Also
      setPinPoint(), setPinPointEnabled()
12.122.4.11 const QPainterPath & QwtSymbol::path ( ) const
Returns
      Painter path for displaying the symbol
See Also
      setPath()
12.122.4.12 const QPen & QwtSymbol::pen ( ) const
Returns
      Pen
See Also
      setPen(), brush()
12.122.4.13 QPointF QwtSymbol::pinPoint ( ) const
Returns
      Pin point
See Also
      setPinPoint(), setPinPointEnabled()
```

12.122.4.14 const QPixmap & QwtSymbol::pixmap () const

Returns

Assigned pixmap

See Also

setPixmap()

12.122.4.15 void QwtSymbol::renderSymbols (QPainter * painter, const QPointF * points, int numPoints) const [protected], [virtual]

Render the symbol to series of points

Parameters

| painter | Qt painter |
|-----------|--------------------------|
| points | Positions of the symbols |
| numPoints | Number of points |

12.122.4.16 void QwtSymbol::setBrush (const QBrush & brush)

Assign a brush.

The brush is used to draw the interior of the symbol.

Parameters

|--|

See Also

brush()

12.122.4.17 void QwtSymbol::setCachePolicy (QwtSymbol::CachePolicy policy)

Change the cache policy

The default policy is AutoCache

Parameters

| policy | Cache policy |
|--------|--------------|

See Also

CachePolicy, cachePolicy()

12.122.4.18 void QwtSymbol::setColor (const QColor & color) [virtual]

Set the color of the symbol.

Change the color of the brush for symbol types with a filled area. For all other symbol types the color will be assigned to the pen.

Parameters

```
color Color
```

See Also

```
setBrush(), setPen(), brush(), pen()
```

12.122.4.19 void QwtSymbol::setGraphic (const QwtGraphic & graphic)

Set a graphic as symbol

Parameters

```
graphic Graphic
```

See Also

```
graphic(), setPixmap()
```

Note

```
the style() is set to QwtSymbol::Graphic brush() and pen() have no effect
```

12.122.4.20 void QwtSymbol::setPath (const QPainterPath & path)

Set a painter path as symbol.

The symbol is represented by a painter path, where the origin (0, 0) of the path coordinate system is mapped to the position of the symbol.

When the symbol has valid size the painter path gets scaled to fit into the size. Otherwise the symbol size depends on the bounding rectangle of the path.

The following code defines a symbol drawing an arrow:

```
#include <qwt_symbol.h>
QwtSymbol *symbol = new QwtSymbol();
QPen pen( Qt::black, 2 );
pen.setJoinStyle( Qt::MiterJoin );
symbol->setPen( pen );
symbol->setBrush( Qt::red );
QPainterPath path;
path.moveTo( 0, 8 );
path.lineTo( 0, 5 );
path.lineTo(-3, 5);
path.lineTo( 0, 0 );
path.lineTo( 3, 5 );
path.lineTo( 0, 5 );
QTransform transform;
transform.rotate( -30.0 );
path = transform.map( path );
symbol->setPath( path );
symbol->setPinPoint( QPointF( 0.0, 0.0 ) );
setSize( 10, 14 );
```

Parameters

| path | Painter path |
|------|--------------|
|------|--------------|

Note

The style is implicitely set to QwtSymbol::Path.

See Also

```
path(), setSize()
```

12.122.4.21 void QwtSymbol::setPen (const QColor & color, qreal width = 0 . 0, Qt::PenStyle style = Qt : : SolidLine)

Build and assign a pen

In Qt5 the default pen width is 1.0 (0.0 in Qt4) what makes it non cosmetic (see QPen::isCosmetic()). This method has been introduced to hide this incompatibility.

Parameters

| color | Pen color |
|-------|-----------|
| width | Pen width |
| style | Pen style |

See Also

pen(), brush()

12.122.4.22 void QwtSymbol::setPen (const QPen & pen)

Assign a pen

The pen is used to draw the symbol's outline.

Parameters

| pen | Pen |
|-----|-----|

See Also

pen(), setBrush()

12.122.4.23 void QwtSymbol::setPinPoint (const QPointF & pos, bool enable = true)

Set and enable a pin point.

The position of a complex symbol is not always aligned to its center (f.e an arrow, where the peak points to a position). The pin point defines the position inside of a Pixmap, Graphic, SvgDocument or PainterPath symbol where the represented point has to be aligned to.

Parameters

| pos | Position |
|--------|------------------------------------|
| enable | En/Disable the pin point alignment |

See Also

pinPoint(), setPinPointEnabled()

12.122.4.24 void QwtSymbol::setPinPointEnabled (bool on)

En/Disable the pin point alignment

Parameters

on Enabled, when on is true

See Also

setPinPoint(), isPinPointEnabled()

12.122.4.25 void QwtSymbol::setPixmap (const QPixmap & pixmap)

Set a pixmap as symbol

Parameters

|--|

See Also

pixmap(), setGraphic()

Note

the style() is set to QwtSymbol::Pixmap brush() and pen() have no effect

12.122.4.26 void QwtSymbol::setSize (const QSize & size)

Set the symbol's size

Parameters

| size | Size |
|------|------|

See Also

size()

12.122.4.27 void QwtSymbol::setSize (int width, int height = -1)

Specify the symbol's size.

If the 'h' parameter is left out or less than 0, and the 'w' parameter is greater than or equal to 0, the symbol size will be set to (w,w).

Parameters

| width | Width |
|--------|-------------------------|
| height | Height (defaults to -1) |

See Also

size()

12.122.4.28 void QwtSymbol::setStyle (QwtSymbol::Style style)

Specify the symbol style

```
Parameters
```

```
style Style
```

See Also

style()

12.122.4.29 void QwtSymbol::setSvgDocument (const QByteArray & svgDocument)

Set a SVG icon as symbol

Parameters

```
svgDocument SVG icon
```

See Also

```
setGraphic(), setPixmap()
```

Note

```
the style() is set to QwtSymbol::SvgDocument brush() and pen() have no effect
```

12.122.4.30 const QSize & QwtSymbol::size () const

Returns

Size

See Also

setSize()

12.122.4.31 QwtSymbol::Style QwtSymbol::style () const

Returns

Current symbol style

See Also

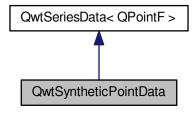
setStyle()

12.123 QwtSyntheticPointData Class Reference

Synthetic point data.

```
#include <qwt_point_data.h>
```

Inheritance diagram for QwtSyntheticPointData:



Public Member Functions

- QwtSyntheticPointData (size_t size, const QwtInterval &=QwtInterval())
- void setSize (size_t size)
- virtual size_t size () const
- void setInterval (const QwtInterval &)
- QwtInterval interval () const
- · virtual QRectF boundingRect () const

Calculate the bounding rectangle.

- virtual QPointF sample (size_t i) const
- virtual double y (double x) const =0
- virtual double x (uint index) const
- virtual void setRectOfInterest (const QRectF &)
- QRectF rectOfInterest () const

Additional Inherited Members

12.123.1 Detailed Description

Synthetic point data.

QwtSyntheticPointData provides a fixed number of points for an interval. The points are calculated in equidistant steps in x-direction.

If the interval is invalid, the points are calculated for the "rectangle of interest", what normally is the displayed area on the plot canvas. In this mode you get different levels of detail, when zooming in/out.

Example

The following example shows how to implement a sinus curve.

```
#include <cmath>
#include <qwt_series_data.h>
#include <qwt_plot_curve.h>
#include <qwt_plot.h>
#include <qapplication.h>
class SinusData: public QwtSyntheticPointData {
  public:
      SinusData():
```

```
QwtSyntheticPointData( 100 )
{
}

virtual double y( double x ) const
{
    return qSin( x );
}

int main(int argc, char **argv)
{
    QApplication a( argc, argv );
    QwtPlot plot;
    plot.setAxisScale( QwtPlot::xBottom, 0.0, 10.0 );
    plot.setAxisScale( QwtPlot::yLeft, -1.0, 1.0 );

    QwtPlotCurve *curve = new QwtPlotCurve( "y = sin(x)" );
    curve->setData( new SinusData() );
    curve->attach( &plot );

    plot.show();
    return a.exec();
}
```

12.123.2 Constructor & Destructor Documentation

12.123.2.1 QwtSyntheticPointData::QwtSyntheticPointData (size_t size, const QwtInterval & interval = QwtInterval ())

Constructor

Parameters

| size | Number of points |
|----------|----------------------------------|
| interval | Bounding interval for the points |

See Also

```
setInterval(), setSize()
```

12.123.3 Member Function Documentation

12.123.3.1 QRectF QwtSyntheticPointData::boundingRect()const [virtual]

Calculate the bounding rectangle.

This implementation iterates over all points, what could often be implemented much faster using the characteristics of the series. When there are many points it is recommended to overload and reimplement this method using the characteristics of the series (if possible).

Returns

Bounding rectangle

Implements QwtSeriesData < QPointF >.

12.123.3.2 QwtInterval QwtSyntheticPointData::interval () const

Returns

Bounding interval

See Also

```
setInterval(), size()
```

```
12.123.3.3 QRectF QwtSyntheticPointData::rectOfInterest ( ) const
Returns
      "rectangle of interest"
See Also
      setRectOfInterest()
12.123.3.4 QPointF QwtSyntheticPointData::sample ( size_t index ) const [virtual]
Calculate the point from an index
Parameters
              index
                      Index
Returns
      QPointF(x(index), y(x(index)));
Warning
      For invalid indices ( index < 0 \mid \mid index >= size() ) (0, 0) is returned.
Implements \ QwtSeriesData < QPointF>.
12.123.3.5 void QwtSyntheticPointData::setInterval ( const QwtInterval & interval )
Set the bounding interval
Parameters
           interval
                      Interval
See Also
      interval(), setSize()
12.123.3.6 void QwtSyntheticPointData::setRectOfInterest ( const QRectF & rect ) [virtual]
Set a the "rectangle of interest"
QwtPlotSeriesItem defines the current area of the plot canvas as "rect of interest" ( QwtPlotSeriesItem::update-
ScaleDiv()).
If interval().isValid() == false the x values are calculated in the interval rect.left() -> rect.right().
See Also
      rectOfInterest()
Reimplemented from QwtSeriesData < QPointF >.
12.123.3.7 void QwtSyntheticPointData::setSize ( size_t size )
Change the number of points
```

Parameters

size Number of points

See Also

size(), setInterval()

12.123.3.8 size_t QwtSyntheticPointData::size() const [virtual]

Returns

Number of points

See Also

setSize(), interval()

Implements QwtSeriesData < QPointF >.

12.123.3.9 double QwtSyntheticPointData::x (uint index) const [virtual]

Calculate a x-value from an index

x values are calculated by dividing an interval into equidistant steps. If !interval().isValid() the interval is calculated from the "rectangle of interest".

Parameters

index Index of the requested point

Returns

Calculated x coordinate

See Also

interval(), rectOfInterest(), y()

12.123.3.10 virtual double QwtSyntheticPointData::y (double x) const [pure virtual]

Calculate a y value for a x value

Parameters

x x value

Returns

Corresponding y value

12.124 QwtSystemClock Class Reference

QwtSystemClock provides high resolution clock time functions.

#include <qwt_system_clock.h>

Public Member Functions

· QwtSystemClock ()

Constructs a null clock object.

virtual ~QwtSystemClock ()

Destructor.

- · bool isNull () const
- void start ()
- · double restart ()
- · double elapsed () const

12.124.1 Detailed Description

QwtSystemClock provides high resolution clock time functions.

Sometimes the resolution offered by QTime (millisecond) is not accurate enough for implementing time measurements (f.e. sampling). QwtSystemClock offers a subset of the QTime functionality using higher resolution timers (if possible).

Precision and time intervals are multiples of milliseconds (ms).

Note

The implementation uses high-resolution performance counter on Windows, mach_absolute_time() on the Mac or POSIX timers on other systems. If none is available it falls back on QTimer.

```
12.124.2 Member Function Documentation
```

12.124.2.1 double QwtSystemClock::elapsed () const

Returns

Number of milliseconds that have elapsed since the last time start() or restart() was called or 0.0 for null clocks.

```
12.124.2.2 bool QwtSystemClock::isNull ( ) const
```

Returns

true if the clock has never been started.

```
12.124.2.3 double QwtSystemClock::restart ( )
```

Set the start time to the current time

Returns

Time, that is elapsed since the previous start time.

```
12.124.2.4 void QwtSystemClock::start ( )
```

Sets the start time to the current time.

12.125 QwtText Class Reference

A class representing a text.

```
#include <qwt_text.h>
```

Public Types

enum TextFormat {
 AutoText = 0, PlainText, RichText, MathMLText,
 TeXText, OtherFormat = 100 }

Text format.

enum PaintAttribute { PaintUsingTextFont = 0x01, PaintUsingTextColor = 0x02, PaintBackground = 0x04 }

Paint Attributes.

enum LayoutAttribute { MinimumLayout = 0x01 }

Layout Attributes The layout attributes affects some aspects of the layout of the text.

typedef QFlags< PaintAttribute > PaintAttributes

Paint attributes.

typedef QFlags< LayoutAttribute > LayoutAttributes

Layout attributes.

Public Member Functions

- QwtText (const QString &=QString::null, TextFormat textFormat=AutoText)
- QwtText (const QwtText &)

Copy constructor.

∼QwtText ()

Destructor.

QwtText & operator= (const QwtText &)

Assignment operator.

bool operator== (const QwtText &) const

Relational operator.

bool operator!= (const QwtText &) const

Relational operator.

- void setText (const QString &, QwtText::TextFormat textFormat=AutoText)
- QString text () const
- · bool isNull () const
- bool isEmpty () const
- void setFont (const QFont &)
- · QFont font () const

Return the font.

- · QFont usedFont (const QFont &) const
- void setRenderFlags (int flags)

Change the render flags.

- int renderFlags () const
- void setColor (const QColor &)
- · QColor color () const

Return the pen color, used for painting the text.

- QColor usedColor (const QColor &) const
- · void setBorderRadius (double)
- · double borderRadius () const
- void setBorderPen (const QPen &)
- QPen borderPen () const
- void setBackgroundBrush (const QBrush &)
- · QBrush backgroundBrush () const
- void setPaintAttribute (PaintAttribute, bool on=true)
- bool testPaintAttribute (PaintAttribute) const
- void setLayoutAttribute (LayoutAttribute, bool on=true)
- bool testLayoutAttribute (LayoutAttribute) const
- double heightForWidth (double width, const QFont &=QFont()) const
- QSizeF textSize (const QFont &=QFont()) const
- · void draw (QPainter *painter, const QRectF &rect) const

Static Public Member Functions

- static const QwtTextEngine * textEngine (const QString &text, QwtText::TextFormat=AutoText)
- static const QwtTextEngine * textEngine (QwtText::TextFormat)

Find the text engine for a text format.

• static void setTextEngine (QwtText::TextFormat, QwtTextEngine *)

12.125.1 Detailed Description

A class representing a text.

A QwtText is a text including a set of attributes how to render it.

Format

A text might include control sequences (f.e tags) describing how to render it. Each format (f.e MathML, TeX, Qt Rich Text) has its own set of control sequences, that can be handles by a special QwtTextEngine for this format.

· Background

A text might have a background, defined by a QPen and QBrush to improve its visibility. The corners of the background might be rounded.

Font

A text might have an individual font.

Color

A text might have an individual color.

· Render Flags

Flags from Qt::AlignmentFlag and Qt::TextFlag used like in QPainter::drawText().

See Also

QwtTextEngine, QwtTextLabel

12.125.2 Member Enumeration Documentation

12.125.2.1 enum QwtText::LayoutAttribute

Layout Attributes The layout attributes affects some aspects of the layout of the text.

Enumerator

MinimumLayout Layout the text without its margins. This mode is useful if a text needs to be aligned accurately, like the tick labels of a scale. If QwtTextEngine::textMargins is not implemented for the format of the text, MinimumLayout has no effect.

12.125.2.2 enum QwtText::PaintAttribute

Paint Attributes.

Font and color and background are optional attributes of a QwtText. The paint attributes hold the information, if they are set.

Enumerator

PaintUsingTextFont The text has an individual font.

PaintUsingTextColor The text has an individual color.

PaintBackground The text has an individual background.

12.125.2.3 enum QwtText::TextFormat

Text format.

The text format defines the QwtTextEngine, that is used to render the text.

See Also

QwtTextEngine, setTextEngine()

Enumerator

AutoText The text format is determined using QwtTextEngine::mightRender() for all available text engines in increasing order > PlainText. If none of the text engines can render the text is rendered like QwtText::-PlainText.

PlainText Draw the text as it is, using a QwtPlainTextEngine.

RichText Use the Scribe framework (Qt Rich Text) to render the text.

```
MathMLText Use a MathML (http://en.wikipedia.org/wiki/MathML) render engine to display the text. The Qwt MathML extension offers such an engine based on the MathML renderer of the Qt solutions package. To enable MathML support the following code needs to be added to the application:
```

QwtText::setTextEngine(QwtText::MathMLText, new QwtMathMLTextEngine());

TeXText Use a TeX (http://en.wikipedia.org/wiki/TeX) render engine to display the text (not implemented yet).

OtherFormat The number of text formats can be extended using setTextEngine. Formats >= QwtText::Other-Format are not used by Qwt.

12.125.3 Constructor & Destructor Documentation

```
12.125.3.1 QwtText::QwtText ( const QString & text = QString::null, QwtText::TextFormat textFormat = AutoText )
```

Constructor

Parameters

| text | Text content |
|------------|--------------|
| textFormat | Text format |

```
12.125.4 Member Function Documentation
```

12.125.4.1 QBrush QwtText::backgroundBrush () const

Returns

Background brush

See Also

setBackgroundBrush(), borderPen()

12.125.4.2 QPen QwtText::borderPen () const

Returns

Background pen

See Also

setBorderPen(), backgroundBrush()

12.125.4.3 double QwtText::borderRadius () const

Returns

Radius for the corners of the border frame

See Also

setBorderRadius(), borderPen(), backgroundBrush()

12.125.4.4 void QwtText::draw (QPainter * painter, const QRectF & rect) const

Draw a text into a rectangle

Parameters

| painter | Painter |
|---------|-----------|
| rect | Rectangle |

12.125.4.5 double QwtText::heightForWidth (double width, const QFont & defaultFont = QFont ()) const

Find the height for a given width

Parameters

| defaultFont | Font, used for the calculation if the text has no font |
|-------------|--|
| width | Width |

Returns

Calculated height

12.125.4.6 bool QwtText::isEmpty () const [inline]

Returns

text().isEmpty()

12.125.4.7 bool QwtText::isNull() const [inline]

Returns

text().isNull()

12.125.4.8 int QwtText::renderFlags () const

Returns

Render flags

See Also

setRenderFlags()

12.125.4.9 void QwtText::setBackgroundBrush (const QBrush & brush)

Set the background brush

Parameters

brush Background brush

See Also

backgroundBrush(), setBorderPen()

12.125.4.10 void QwtText::setBorderPen (const QPen & pen)

Set the background pen

Parameters

pen Background pen

See Also

borderPen(), setBackgroundBrush()

12.125.4.11 void QwtText::setBorderRadius (double radius)

Set the radius for the corners of the border frame

Parameters

radius Radius of a rounded corner

See Also

borderRadius(), setBorderPen(), setBackgroundBrush()

12.125.4.12 void QwtText::setColor (const QColor & color)

Set the pen color used for drawing the text.

Parameters

color Color

Note

Setting the color might have no effect, when the text contains control sequences for setting colors.

12.125.4.13 void QwtText::setFont (const QFont & font)

Set the font.

Parameters

font Font

Note

Setting the font might have no effect, when the text contains control sequences for setting fonts.

12.125.4.14 void QwtText::setLayoutAttribute (LayoutAttribute attribute, bool on = true)

Change a layout attribute

Parameters

| attribute | Layout attribute |
|-----------|------------------|
| on | On/Off |

See Also

testLayoutAttribute()

12.125.4.15 void QwtText::setPaintAttribute (PaintAttribute attribute, bool on = true)

Change a paint attribute

Parameters

| attribute | Paint attribute |
|-----------|-----------------|
| on | On/Off |

Note

Used by setFont(), setColor(), setBorderPen() and setBackgroundBrush()

See Also

testPaintAttribute()

12.125.4.16 void QwtText::setRenderFlags (int renderFlags)

Change the render flags.

The default setting is Qt::AlignCenter

Parameters

| renderFlags | Bitwise OR of the flags used like in QPainter::drawText() |
|-------------|---|
| | , , |

See Also

renderFlags(), QwtTextEngine::draw()

Note

Some renderFlags might have no effect, depending on the text format.

12.125.4.17 void QwtText::setText (const QString & text, QwtText::TextFormat textFormat = AutoText)

Assign a new text content

Parameters

| text | Text content |
|------------|--------------|
| textFormat | Text format |

See Also

text()

12.125.4.18 void QwtText::setTextEngine (QwtText::TextFormat format, QwtTextEngine * engine) [static]

Assign/Replace a text engine for a text format

With setTextEngine it is possible to extend Qwt with other types of text formats.

For QwtText::PlainText it is not allowed to assign a engine == NULL.

| format | Text format |
|--------|-------------|
| engine | Text engine |

See Also

QwtMathMLTextEngine

Warning

Using QwtText::AutoText does nothing.

12.125.4.19 bool QwtText::testLayoutAttribute (LayoutAttribute attribute) const

Test a layout attribute

Parameters

| attribute | Layout attribute |
|-----------|------------------|

Returns

true, if attribute is enabled

See Also

setLayoutAttribute()

12.125.4.20 bool QwtText::testPaintAttribute (PaintAttribute attribute) const

Test a paint attribute

Parameters

| attril | Paint attribute |
|--------|-----------------|
|--------|-----------------|

Returns

true, if attribute is enabled

See Also

setPaintAttribute()

12.125.4.21 QString QwtText::text () const

Returns

Text as QString.

See Also

setText()

12.125.4.22 const QwtTextEngine * QwtText::textEngine (const QString & text, QwtText::TextFormat format = AutoText) [static]

Find the text engine for a text format

In case of QwtText::AutoText the first text engine (beside QwtPlainTextEngine) is returned, where QwtTextEngine::mightRender returns true. If there is none QwtPlainTextEngine is returned.

If no text engine is registered for the format QwtPlainTextEngine is returnd.

Parameters

| text | Text, needed in case of AutoText |
|--------|----------------------------------|
| format | Text format |

Returns

Corresponding text engine

12.125.4.23 const QwtTextEngine * QwtText::textEngine (QwtText::TextFormat format) [static]

Find the text engine for a text format.

textEngine can be used to find out if a text format is supported.

Parameters

| format | Text format |
|--------|-------------|

Returns

The text engine, or NULL if no engine is available.

12.125.4.24 QSizeF QwtText::textSize (const QFont & defaultFont = QFont ()) const

Find the height for a given width

Parameters

| _ | | |
|---|-------------|--|
| | defaultFont | Font, used for the calculation if the text has no font |

Returns

Calculated height

Returns the size, that is needed to render text

Parameters

| defaultFont | Font of the text |
|-------------|------------------|

Returns

Caluclated size

12.125.4.25 QColor QwtText::usedColor (const QColor & defaultColor) const

Return the color of the text, if it has one. Otherwise return defaultColor.

Parameters

| defaultColor | Default color |
|--------------|---------------|

Returns

Color used for drawing the text

See Also

setColor(), color(), PaintAttributes

12.125.4.26 QFont QwtText::usedFont (const QFont & defaultFont) const

Return the font of the text, if it has one. Otherwise return defaultFont.

| defaultFont | Default font |
|-------------|--------------|
|-------------|--------------|

Returns

Font used for drawing the text

See Also

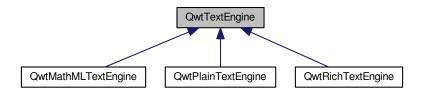
setFont(), font(), PaintAttributes

12.126 QwtTextEngine Class Reference

Abstract base class for rendering text strings.

#include <qwt_text_engine.h>

Inheritance diagram for QwtTextEngine:



Public Member Functions

- virtual ~QwtTextEngine ()
 - Destructor.
- virtual double heightForWidth (const QFont &font, int flags, const QString &text, double width) const =0
- virtual QSizeF textSize (const QFont &font, int flags, const QString &text) const =0
- virtual bool mightRender (const QString &text) const =0
- virtual void textMargins (const QFont &font, const QString &text, double &left, double &right, double &top, double &bottom) const =0
- virtual void draw (QPainter *painter, const QRectF &rect, int flags, const QString &text) const =0

Protected Member Functions

QwtTextEngine ()

Constructor.

12.126.1 Detailed Description

Abstract base class for rendering text strings.

A text engine is responsible for rendering texts for a specific text format. They are used by QwtText to render a text.

QwtPlainTextEngine and QwtRichTextEngine are part of the Qwt library. The implementation of QwtMathMLText-Engine uses code from the Qt solution package. Because of license implications it is built into a separate library.

See Also

QwtText::setTextEngine()

12.126.2 Member Function Documentation

12.126.2.1 virtual void QwtTextEngine::draw (QPainter * painter, const QRectF & rect, int flags, const QString & text) const [pure virtual]

Draw the text in a clipping rectangle

Parameters

| painter | Painter |
|---------|--|
| rect | Clipping rectangle |
| flags | Bitwise OR of the flags like in for QPainter::drawText() |
| text | Text to be rendered |

Implemented in QwtRichTextEngine, QwtPlainTextEngine, and QwtMathMLTextEngine.

12.126.2.2 virtual double QwtTextEngine::heightForWidth (const QFont & font, int flags, const QString & text, double width)

const [pure virtual]

Find the height for a given width

Parameters

| font | Font of the text |
|-------|---|
| flags | Bitwise OR of the flags used like in QPainter::drawText |
| text | Text to be rendered |
| width | Width |

Returns

Calculated height

Implemented in QwtRichTextEngine, QwtPlainTextEngine, and QwtMathMLTextEngine.

12.126.2.3 virtual bool QwtTextEngine::mightRender (const QString & text) const [pure virtual]

Test if a string can be rendered by this text engine

Parameters

| text | Text to be tested |
|------|-------------------|

Returns

true, if it can be rendered

Implemented in QwtRichTextEngine, QwtPlainTextEngine, and QwtMathMLTextEngine.

12.126.2.4 virtual void QwtTextEngine::textMargins (const QFont & font, const QString & text, double & left, double & right, double & top, double & bottom) const [pure virtual]

Return margins around the texts

The textSize might include margins around the text, like QFontMetrics::descent(). In situations where texts need to be aligned in detail, knowing these margins might improve the layout calculations.

| font | Font of the text |
|--------|------------------------------------|
| text | Text to be rendered |
| left | Return value for the left margin |
| right | Return value for the right margin |
| top | Return value for the top margin |
| bottom | Return value for the bottom margin |

Implemented in QwtRichTextEngine, QwtPlainTextEngine, and QwtMathMLTextEngine.

12.126.2.5 virtual QSizeF QwtTextEngine::textSize (const QFont & font, int flags, const QString & text) const [pure virtual]

Returns the size, that is needed to render text

Parameters

| font | Font of the text |
|-------|--|
| flags | Bitwise OR of the flags like in for QPainter::drawText |
| text | Text to be rendered |

Returns

Calculated size

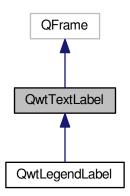
Implemented in QwtRichTextEngine, QwtPlainTextEngine, and QwtMathMLTextEngine.

12.127 QwtTextLabel Class Reference

A Widget which displays a QwtText.

#include <qwt_text_label.h>

Inheritance diagram for QwtTextLabel:



Public Slots

- void setText (const QString &, QwtText::TextFormat textFormat=QwtText::AutoText)
- virtual void setText (const QwtText &)

• void clear ()

Clear the text and all QwtText attributes.

Public Member Functions

- QwtTextLabel (QWidget *parent=NULL)
- QwtTextLabel (const QwtText &, QWidget *parent=NULL)
- virtual ~QwtTextLabel ()

Destructor.

- void setPlainText (const QString &)
- QString plainText () const
- const QwtText & text () const

Return the text.

· int indent () const

Return label's text indent in pixels.

- · void setIndent (int)
- int margin () const

Return label's text indent in pixels.

- void setMargin (int)
- virtual QSize sizeHint () const

Return label's margin in pixels.

• virtual QSize minimumSizeHint () const

Return a minimum size hint.

- virtual int heightForWidth (int) const
- QRect textRect () const
- virtual void drawText (QPainter *, const QRectF &)

Redraw the text.

Protected Member Functions

- virtual void paintEvent (QPaintEvent *e)
- virtual void drawContents (QPainter *)

Redraw the text and focus indicator.

12.127.1 Detailed Description

A Widget which displays a QwtText.

12.127.2 Constructor & Destructor Documentation

12.127.2.1 QwtTextLabel::QwtTextLabel(QWidget * parent = NULL) [explicit]

Constructs an empty label.

Parameters

parent | Parent widget

12.127.2.2 QwtTextLabel::QwtTextLabel(const QwtText & text, QWidget * parent = NULL) [explicit]

Constructs a label that displays the text, text

| parent | Parent widget |
|--------|---------------|
| text | Text |

12.127.3 Member Function Documentation

12.127.3.1 int QwtTextLabel::heightForWidth (int width) const [virtual]

Parameters

| width | Width |
|-------|-------|

Returns

Preferred height for this widget, given the width.

12.127.3.2 void QwtTextLabel::paintEvent (QPaintEvent * event) [protected], [virtual]

Qt paint event

Parameters

| event | Paint event |
|-------|-------------|

Reimplemented in QwtLegendLabel.

12.127.3.3 QString QwtTextLabel::plainText () const

Interface for the designer plugin

Returns

Text as plain text

See Also

setPlainText(), text()

12.127.3.4 void QwtTextLabel::setIndent (int indent)

Set label's text indent in pixels

Parameters

| indent | Indentation in pixels |
|--------|-----------------------|

12.127.3.5 void QwtTextLabel::setMargin (int margin)

Set label's margin in pixels

Parameters

| margin | Margin in pixels |
|--------|------------------|

12.127.3.6 void QwtTextLabel::setPlainText (const QString & text)

Interface for the designer plugin - does the same as setText()

See Also

plainText()

12.127.3.7 void QwtTextLabel::setText (const QString & text, QwtText::TextFormat textFormat = QwtText::AutoText) [slot]

Change the label's text, keeping all other QwtText attributes

| text | New text |
|------------|----------------|
| textFormat | Format of text |

See Also

QwtText

12.127.3.8 void QwtTextLabel::setText (const QwtText & text) [virtual], [slot]

Change the label's text

Parameters

| text | New text |
|------|----------|
| | |

Reimplemented in QwtLegendLabel.

12.127.3.9 QRect QwtTextLabel::textRect () const

Calculate geometry for the text in widget coordinates

Returns

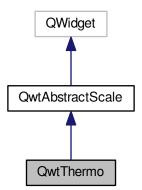
Geometry for the text

12.128 QwtThermo Class Reference

The Thermometer Widget.

#include <qwt_thermo.h>

Inheritance diagram for QwtThermo:



Public Types

- enum ScalePosition { NoScale, LeadingScale, TrailingScale }
- enum OriginMode { OriginMinimum, OriginMaximum, OriginCustom }

Public Slots

• virtual void setValue (double val)

Public Member Functions

- QwtThermo (QWidget *parent=NULL)
- virtual ~QwtThermo ()

Destructor.

void setOrientation (Qt::Orientation)

Set the orientation.

- · Qt::Orientation orientation () const
- void setScalePosition (ScalePosition)

Change the position of the scale.

- ScalePosition scalePosition () const
- void setSpacing (int)

Change the spacing between pipe and scale.

- int spacing () const
- void setBorderWidth (int w)
- int borderWidth () const
- void setOriginMode (OriginMode)

Change how the origin is determined.

- OriginMode originMode () const
- void setOrigin (double)

Specifies the custom origin.

- · double origin () const
- void setFillBrush (const QBrush &b)

Change the brush of the liquid.

- QBrush fillBrush () const
- void setAlarmBrush (const QBrush &b)

Specify the liquid brush above the alarm threshold.

- QBrush alarmBrush () const
- void setAlarmLevel (double v)
- double alarmLevel () const
- void setAlarmEnabled (bool tf)

Enable or disable the alarm threshold.

- bool alarmEnabled () const
- void setColorMap (QwtColorMap *)

Assign a color map for the fill color.

- QwtColorMap * colorMap ()
- const QwtColorMap * colorMap () const
- void setPipeWidth (int w)
- int pipeWidth () const
- void setRangeFlags (QwtInterval::BorderFlags)

Exclude/Include min/max values.

- QwtInterval::BorderFlags rangeFlags () const
- double value () const

Return the value.

- · virtual QSize sizeHint () const
- virtual QSize minimumSizeHint () const
- void setScaleDraw (QwtScaleDraw *)

Set a scale draw.

const QwtScaleDraw * scaleDraw () const

Protected Member Functions

- virtual void drawLiquid (QPainter *, const QRect &) const
- virtual void scaleChange ()

Notify a scale change.

- virtual void paintEvent (QPaintEvent *)
- virtual void resizeEvent (QResizeEvent *)
- virtual void changeEvent (QEvent *)
- QwtScaleDraw * scaleDraw ()
- QRect pipeRect () const
- QRect fillRect (const QRect &) const

Calculate the filled rectangle of the pipe.

QRect alarmRect (const QRect &) const

Calculate the alarm rectangle of the pipe.

12.128.1 Detailed Description

The Thermometer Widget.

QwtThermo is a widget which displays a value in an interval. It supports:

- · a horizontal or vertical layout;
- · a range;
- · a scale;
- · an alarm level.

The fill colors might be calculated from an optional color map If no color map has been assigned QwtThermo uses the following colors/brushes from the widget palette:

- · QPalette::Base Background of the pipe
- · QPalette::ButtonText Fill brush below the alarm level
- · QPalette::Highlight Fill brush for the values above the alarm level
- · QPalette::WindowText For the axis of the scale
- · QPalette::Text For the labels of the scale

12.128.2 Member Enumeration Documentation

12.128.2.1 enum QwtThermo::OriginMode

Origin mode. This property specifies where the beginning of the liquid is placed.

See Also

```
setOriginMode(), setOrigin()
```

Enumerator

OriginMinimum The origin is the minimum of the scale.

OriginMaximum The origin is the maximum of the scale.

OriginCustom The origin is specified using the origin() property.

12.128.2.2 enum QwtThermo::ScalePosition

Position of the scale

See Also

setScalePosition(), setOrientation()

Enumerator

NoScale The slider has no scale.

LeadingScale The scale is right of a vertical or below of a horizontal slider.

TrailingScale The scale is left of a vertical or above of a horizontal slider.

12.128.3 Constructor & Destructor Documentation

12.128.3.1 QwtThermo::QwtThermo(QWidget*parent=NULL) [explicit]

Constructor

Parameters

parent Parent widget

12.128.4 Member Function Documentation

12.128.4.1 QBrush QwtThermo::alarmBrush () const

Returns

Liquid brush (QPalette::Highlight) above the alarm threshold.

See Also

setAlarmBrush(), QWidget::palette()

Warning

The alarm threshold has no effect, when a color map has been assigned

12.128.4.2 bool QwtThermo::alarmEnabled () const

Returns

True, when the alarm threshold is enabled.

Warning

The alarm threshold has no effect, when a color map has been assigned

12.128.4.3 double QwtThermo::alarmLevel () const

Returns

Alarm threshold.

```
See Also
```

setAlarmLevel()

Warning

The alarm threshold has no effect, when a color map has been assigned

12.128.4.4 QRect QwtThermo::alarmRect (const QRect & fillRect) const [protected]

Calculate the alarm rectangle of the pipe.

Parameters

fillRect | Filled rectangle in the pipe

Returns

Rectangle to be filled with the alarm brush

See Also

```
pipeRect(), fillRect(), alarmLevel(), alarmBrush()
```

12.128.4.5 int QwtThermo::borderWidth () const

Returns

Border width of the thermometer pipe.

See Also

setBorderWidth()

12.128.4.6 void QwtThermo::changeEvent (QEvent * event) [protected], [virtual]

Qt change event handler

Parameters

event Event

12.128.4.7 QwtColorMap * QwtThermo::colorMap ()

Returns

Color map for the fill color

Warning

The alarm threshold has no effect, when a color map has been assigned

12.128.4.8 const QwtColorMap * QwtThermo::colorMap () const

Returns

Color map for the fill color

Warning

The alarm threshold has no effect, when a color map has been assigned

12.128.4.9 void QwtThermo::drawLiquid (QPainter * painter, const QRect & pipeRect) const [protected], [virtual]

Redraw the liquid in thermometer pipe.

| painter | Painter |
|----------|--|
| pipeRect | Bounding rectangle of the pipe without borders |

12.128.4.10 QBrush QwtThermo::fillBrush () const

Returns

Liquid (QPalette::ButtonText) brush.

See Also

```
setFillBrush(), QWidget::palette()
```

12.128.4.11 QRect QwtThermo::fillRect (const QRect & pipeRect) const [protected]

Calculate the filled rectangle of the pipe.

Parameters

| pipeRect | Rectangle of the pipe |
|----------|-----------------------|
|----------|-----------------------|

Returns

Rectangle to be filled (fill and alarm brush)

See Also

pipeRect(), alarmRect()

12.128.4.12 QSize QwtThermo::minimumSizeHint() const [virtual]

Returns

Minimum size hint

Warning

The return value depends on the font and the scale.

See Also

sizeHint()

12.128.4.13 Qt::Orientation QwtThermo::orientation () const

Returns

Orientation

See Also

setOrientation()

```
12.128.4.14 double QwtThermo::origin ( ) const
Returns
     Origin of the thermo, when OriginCustom is enabled
See Also
     setOrigin(), setOriginMode(), originMode()
12.128.4.15 QwtThermo::OriginMode QwtThermo::originMode ( ) const
Returns
     Mode, how the origin is determined.
See Also
     setOriginMode(), serOrigin(), origin()
12.128.4.16 void QwtThermo::paintEvent ( QPaintEvent * event ) [protected], [virtual]
Paint event handler
Parameters
             event
                     Paint event
12.128.4.17 QRect QwtThermo::pipeRect( ) const [protected]
Returns
     Bounding rectangle of the pipe (without borders) in widget coordinates
12.128.4.18 int QwtThermo::pipeWidth ( ) const
Returns
     Width of the pipe.
See Also
     setPipeWidth()
12.128.4.19 QwtInterval::BorderFlags QwtThermo::rangeFlags ( ) const
Returns
     Range flags
See Also
     setRangeFlags()
12.128.4.20 void QwtThermo::resizeEvent ( QResizeEvent * event ) [protected], [virtual]
Resize event handler
```

```
Parameters
```

```
event
                     Resize event
12.128.4.21 const QwtScaleDraw * QwtThermo::scaleDraw ( ) const
Returns
     the scale draw of the thermo
See Also
     setScaleDraw()
12.128.4.22 QwtScaleDraw * QwtThermo::scaleDraw() [protected]
Returns
     the scale draw of the thermo
See Also
     setScaleDraw()
12.128.4.23 QwtThermo::ScalePosition QwtThermo::scalePosition ( ) const
Returns
     Scale position.
See Also
     setScalePosition()
12.128.4.24 void QwtThermo::setAlarmBrush ( const QBrush & brush )
Specify the liquid brush above the alarm threshold.
Changes the QPalette::Highlight brush of the palette.
Parameters
             brush
                     New brush.
See Also
     alarmBrush(), QWidget::setPalette()
Warning
     The alarm threshold has no effect, when a color map has been assigned
12.128.4.25 void QwtThermo::setAlarmEnabled (bool on)
Enable or disable the alarm threshold.
```

Parameters

| on | true (disabled) or false (enabled) |
|----|------------------------------------|

Warning

The alarm threshold has no effect, when a color map has been assigned

12.128.4.26 void QwtThermo::setAlarmLevel (double level)

Specify the alarm threshold.

Parameters

| level Alarm threshold | |
|-----------------------|--|
|-----------------------|--|

See Also

alarmLevel()

Warning

The alarm threshold has no effect, when a color map has been assigned

12.128.4.27 void QwtThermo::setBorderWidth (int width)

Set the border width of the pipe.

Parameters

| width | Border width |
|-------|--------------|
|-------|--------------|

See Also

borderWidth()

12.128.4.28 void QwtThermo::setColorMap (QwtColorMap * colorMap)

Assign a color map for the fill color.

Parameters

| colorMap | Color map |
|----------|-----------|

Warning

The alarm threshold has no effect, when a color map has been assigned

12.128.4.29 void QwtThermo::setFillBrush (const QBrush & brush)

Change the brush of the liquid.

Changes the QPalette::ButtonText brush of the palette.

Parameters

brush New brush.

See Also

fillBrush(), QWidget::setPalette()

12.128.4.30 void QwtThermo::setOrientation (Qt::Orientation orientation)

Set the orientation.

Parameters

orientation Allowed values are Qt::Horizontal and Qt::Vertical.

See Also

orientation(), scalePosition()

12.128.4.31 void QwtThermo::setOrigin (double origin)

Specifies the custom origin.

If originMode is set to OriginCustom this property controls where the liquid starts.

Parameters

origin New origin level

See Also

setOriginMode(), originMode(), origin()

12.128.4.32 void QwtThermo::setOriginMode (OriginMode m)

Change how the origin is determined.

See Also

originMode(), serOrigin(), origin()

12.128.4.33 void QwtThermo::setPipeWidth (int width)

Change the width of the pipe.

Parameters

width Width of the pipe

See Also

pipeWidth()

12.128.4.34 void QwtThermo::setRangeFlags (QwtInterval::BorderFlags flags)

Exclude/Include min/max values.

According to the flags minValue() and maxValue() are included/excluded from the pipe. In case of an excluded value the corresponding tick is painted 1 pixel off of the pipeRect().

F.e. when a minimum of 0.0 has to be displayed as an empty pipe the minValue() needs to be excluded.

Parameters

flags Range flags

See Also

rangeFlags()

12.128.4.35 void QwtThermo::setScaleDraw (QwtScaleDraw * scaleDraw)

Set a scale draw.

For changing the labels of the scales, it is necessary to derive from QwtScaleDraw and overload QwtScaleDraw::label().

Parameters

scaleDraw ScaleDraw object, that has to be created with new and will be deleted in ~QwtThermo() or the next call of setScaleDraw().

12.128.4.36 void QwtThermo::setScalePosition (ScalePosition scalePosition)

Change the position of the scale.

Parameters

scalePosition Position of the scale.

See Also

ScalePosition, scalePosition()

12.128.4.37 void QwtThermo::setSpacing (int spacing)

Change the spacing between pipe and scale.

A spacing of 0 means, that the backbone of the scale is below the pipe.

The default setting is 3 pixels.

Parameters

spacing Number of pixels

See Also

spacing();

12.128.4.38 void QwtThermo::setValue (double value) [virtual], [slot]

Set the current value.

Parameters

value New Value

See Also

value()

12.128.4.39 QSize QwtThermo::sizeHint() const [virtual]

Returns

the minimum size hint

See Also

minimumSizeHint()

12.128.4.40 int QwtThermo::spacing () const

Returns

Number of pixels between pipe and scale

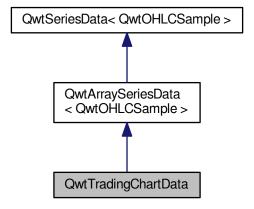
See Also

setSpacing()

12.129 QwtTradingChartData Class Reference

#include <qwt_series_data.h>

Inheritance diagram for QwtTradingChartData:



Public Member Functions

- QwtTradingChartData (const QVector< QwtOHLCSample > &=QVector< QwtOHLCSample >())
- virtual QRectF boundingRect () const Calculate the bounding rectangle.

Additional Inherited Members

12.129.1 Detailed Description

Interface for iterating over an array of OHLC samples

12.129.2 Constructor & Destructor Documentation

12.129.2.1 QwtTradingChartData::QwtTradingChartData (const QVector< QwtOHLCSample > & samples = QVector<QwtOHLCSample>())

Constructor

Parameters

| samples | Samples |
|---------|---------|

12.129.3 Member Function Documentation

12.129.3.1 QRectF QwtTradingChartData::boundingRect() const [virtual]

Calculate the bounding rectangle.

The bounding rectangle is calculated once by iterating over all points and is stored for all following requests.

Returns

Bounding rectangle

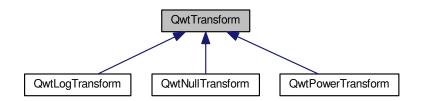
Implements QwtSeriesData < QwtOHLCSample >.

12.130 QwtTransform Class Reference

A transformation between coordinate systems.

#include <qwt_transform.h>

Inheritance diagram for QwtTransform:



Public Member Functions

• QwtTransform ()

Constructor.

virtual ~QwtTransform ()

Destructor.

- virtual double bounded (double value) const
- virtual double transform (double value) const =0
- virtual double invTransform (double value) const =0
- virtual QwtTransform * copy () const =0

Virtualized copy operation.

12.130.1 Detailed Description

A transformation between coordinate systems.

QwtTransform manipulates values, when being mapped between the scale and the paint device coordinate system.

A transformation consists of 2 methods:

- · transform
- · invTransform

where one is is the inverse function of the other.

When p1, p2 are the boundaries of the paint device coordinates and s1, s2 the boundaries of the scale, QwtScale-Map uses the following calculations:

```
• p = p1 + (p2 - p1) * (T(s) - T(s1) / (T(s2) - T(s1));
```

•
$$s = invT (T(s1) + (T(s2) - T(s1)) * (p - p1) / (p2 - p1));$$

12.130.2 Member Function Documentation

12.130.2.1 double QwtTransform::bounded (double value) const [virtual]

Modify value to be a valid value for the transformation. The default implementation does nothing.

Parameters

| value | Value to be bounded |
|-------|---------------------|
| Value | value to be bearied |

Returns

value unmodified

Reimplemented in QwtLogTransform.

12.130.2.2 virtual double QwtTransform::invTransform (double value) const [pure virtual]

Inverse transformation function

Parameters

| value | Value |
|-------|-------|

Returns

Modified value

See Also

transform()

Implemented in QwtPowerTransform, QwtLogTransform, and QwtNullTransform.

12.130.2.3 virtual double QwtTransform::transform (double value) const [pure virtual]

Transformation function

Parameters

| value | Value |
|-------|-------|
|-------|-------|

Returns

Modified value

See Also

invTransform()

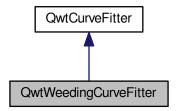
Implemented in QwtPowerTransform, QwtLogTransform, and QwtNullTransform.

12.131 QwtWeedingCurveFitter Class Reference

A curve fitter implementing Douglas and Peucker algorithm.

```
#include <qwt_curve_fitter.h>
```

Inheritance diagram for QwtWeedingCurveFitter:



Public Member Functions

- QwtWeedingCurveFitter (double tolerance=1.0)
- virtual ~QwtWeedingCurveFitter ()

Destructor.

- void setTolerance (double)
- · double tolerance () const
- void setChunkSize (uint)
- uint chunkSize () const
- virtual QPolygonF fitCurve (const QPolygonF &) const

Additional Inherited Members

12.131.1 Detailed Description

A curve fitter implementing Douglas and Peucker algorithm.

The purpose of the Douglas and Peucker algorithm is that given a 'curve' composed of line segments to find a curve not too dissimilar but that has fewer points. The algorithm defines 'too dissimilar' based on the maximum distance (tolerance) between the original curve and the smoothed curve.

The runtime of the algorithm increases non linear (worst case O(n*n)) and might be very slow for huge polygons. To avoid performance issues it might be useful to split the polygon (setChunkSize()) and to run the algorithm for these smaller parts. The disadvantage of having no interpolation at the borders is for most use cases irrelevant.

The smoothed curve consists of a subset of the points that defined the original curve.

In opposite to QwtSplineCurveFitter the Douglas and Peucker algorithm reduces the number of points. By adjusting the tolerance parameter according to the axis scales QwtSplineCurveFitter can be used to implement different level of details to speed up painting of curves of many points.

12.131.2 Constructor & Destructor Documentation

12.131.2.1 QwtWeedingCurveFitter::QwtWeedingCurveFitter (double tolerance = 1 . 0)

Constructor

Parameters

tolerance Tolerance

See Also

setTolerance(), tolerance()

12.131.3 Member Function Documentation

12.131.3.1 uint QwtWeedingCurveFitter::chunkSize () const

Returns

Maximum for the number of points passed to a run of the algorithm - or 0, when unlimited

See Also

setChunkSize()

12.131.3.2 QPolygonF QwtWeedingCurveFitter::fitCurve (const QPolygonF & points) const [virtual]

Parameters

points | Series of data points

Returns

Curve points

Implements QwtCurveFitter.

12.131.3.3 void QwtWeedingCurveFitter::setChunkSize (uint numPoints)

Limit the number of points passed to a run of the algorithm

The runtime of the Douglas Peucker algorithm increases non linear with the number of points. For a chunk size > 0 the polygon is split into pieces passed to the algorithm one by one.

Parameters

numPoints | Maximum for the number of points passed to the algorithm

See Also

chunkSize()

12.131.3.4 void QwtWeedingCurveFitter::setTolerance (double tolerance)

Assign the tolerance

The tolerance is the maximum distance, that is acceptable between the original curve and the smoothed curve.

Increasing the tolerance will reduce the number of the resulting points.

Parameters

tolerance Tolerance

See Also

tolerance()

12.131.3.5 double QwtWeedingCurveFitter::tolerance () const

Returns

Tolerance

See Also

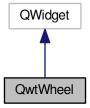
setTolerance()

12.132 QwtWheel Class Reference

The Wheel Widget.

#include <qwt_wheel.h>

Inheritance diagram for QwtWheel:



Public Slots

void setValue (double)

Set a new value without adjusting to the step raster.

void setTotalAngle (double)

Set the total angle which the wheel can be turned.

void setViewAngle (double)

Specify the visible portion of the wheel.

void setMass (double)

Set the slider's mass for flywheel effect.

Signals

· void valueChanged (double value)

Notify a change of value.

- · void wheelPressed ()
- void wheelReleased ()
- void wheelMoved (double value)

Public Member Functions

QwtWheel (QWidget *parent=NULL)

Constructor.

• virtual \sim QwtWheel ()

Destructor.

- double value () const
- void setOrientation (Qt::Orientation)

Set the wheel's orientation.

- Qt::Orientation orientation () const
- double totalAngle () const
- double viewAngle () const
- · void setTickCount (int)

Adjust the number of grooves in the wheel's surface.

- · int tickCount () const
- void setWheelWidth (int)

Set the width of the wheel.

- int wheelWidth () const
- void setWheelBorderWidth (int)

Set the wheel border width of the wheel.

- int wheelBorderWidth () const
- void setBorderWidth (int)

Set the border width.

- int borderWidth () const
- void setInverted (bool tf)

En/Disable inverted appearance.

- · bool isInverted () const
- void setWrapping (bool tf)

En/Disable wrapping.

- bool wrapping () const
- void setSingleStep (double)

Set the step size of the counter.

- double singleStep () const
- void setPageStepCount (int)

Set the page step count.

• int pageStepCount () const

void setStepAlignment (bool on)

En/Disable step alignment.

- bool stepAlignment () const
- void setRange (double vmin, double vmax)

Set the minimum and maximum values.

- void setMinimum (double min)
- double minimum () const
- void setMaximum (double max)
- · double maximum () const
- void setUpdateInterval (int)

Specify the update interval when the wheel is flying.

- int updateInterval () const
- void setTracking (bool enable)

En/Disable tracking.

- · bool isTracking () const
- · double mass () const

Protected Member Functions

virtual void paintEvent (QPaintEvent *)

Qt Paint Event.

virtual void mousePressEvent (QMouseEvent *)

Mouse press event handler.

virtual void mouseReleaseEvent (QMouseEvent *)

Mouse Release Event handler.

virtual void mouseMoveEvent (QMouseEvent *)

Mouse Move Event handler.

- virtual void keyPressEvent (QKeyEvent *)
- virtual void wheelEvent (QWheelEvent *)

Handle wheel events.

virtual void timerEvent (QTimerEvent *)

Qt timer event.

· void stopFlying ()

Stop the flying movement of the wheel.

- QRect wheelRect () const
- virtual QSize sizeHint () const
- virtual QSize minimumSizeHint () const
- virtual void drawTicks (QPainter *, const QRectF &)
- virtual void drawWheelBackground (QPainter *, const QRectF &)
- · virtual double valueAt (const QPoint &) const

12.132.1 Detailed Description

The Wheel Widget.

The wheel widget can be used to change values over a very large range in very small steps. Using the setMass() member, it can be configured as a flying wheel.

The default range of the wheel is [0.0, 100.0]

See Also

The radio example.

12.132.2 Member Function Documentation

12.132.2.1 int QwtWheel::borderWidth () const

Returns

Border width

See Also

setBorderWidth()

12.132.2.2 void QwtWheel::drawTicks (QPainter * painter, const QRectF & rect) [protected], [virtual]

Draw the Wheel's ticks

Parameters

| painter | Painter |
|---------|------------------------|
| rect | Geometry for the wheel |

12.132.2.3 void QwtWheel::drawWheelBackground (QPainter * *painter*, **const QRectF &** *rect*) [protected], [virtual]

Draw the Wheel's background gradient

Parameters

| painter | Painter |
|---------|------------------------|
| rect | Geometry for the wheel |

12.132.2.4 bool QwtWheel::isInverted () const

Returns

True, when the wheel is inverted

See Also

setInverted()

12.132.2.5 bool QwtWheel::isTracking () const

Returns

True, when tracking is enabled

See Also

setTracking(), valueChanged(), wheelMoved()

12.132.2.6 void QwtWheel::keyPressEvent (QKeyEvent * event) [protected], [virtual]

Handle key events

Qt::Key_Home
 Step to minimum()

Qt::Key_End
 Step to maximum()

· Qt::Key_Up

In case of a horizontal or not inverted vertical wheel the value will be incremented by the step size. For an inverted vertical wheel the value will be decremented by the step size.

· Qt::Key_Down

In case of a horizontal or not inverted vertical wheel the value will be decremented by the step size. For an inverted vertical wheel the value will be incremented by the step size.

· Qt::Key_PageUp

The value will be incremented by pageStepSize() * singleStepSize().

· Qt::Key_PageDown

Turn the wheel according to the mouse position

The value will be decremented by pageStepSize() * singleStepSize().

Parameters

```
event
                    Key event
12.132.2.7 double QwtWheel::mass ( ) const
Returns
     mass
See Also
     setMass()
12.132.2.8 double QwtWheel::maximum ( ) const
Returns
     The maximum of the range
See Also
     setRange(), setMaximum(), minimum()
12.132.2.9 double QwtWheel::minimum ( ) const
Returns
     The minimum of the range
See Also
     setRange(), setMinimum(), maximum()
12.132.2.10 QSize QwtWheel::minimumSizeHint() const [protected], [virtual]
Returns
     Minimum size hint
Warning
     The return value is based on the wheel width.
12.132.2.11 void QwtWheel::mouseMoveEvent( QMouseEvent * event) [protected], [virtual]
Mouse Move Event handler.
```

event Mouse event

 $\textbf{12.132.2.12} \quad \textbf{void QwtWheel::mousePressEvent (QMouseEvent} * \textit{event} \;) \quad \texttt{[protected], [virtual]}$

Mouse press event handler.

Start movement of the wheel.

Parameters

event Mouse event

12.132.2.13 void QwtWheel::mouseReleaseEvent (QMouseEvent * event) [protected], [virtual]

Mouse Release Event handler.

When the wheel has no mass the movement of the wheel stops, otherwise it starts flying.

Parameters

event Mouse event

12.132.2.14 Qt::Orientation QwtWheel::orientation () const

Returns

Orientation

See Also

setOrientation()

12.132.2.15 int QwtWheel::pageStepCount () const

Returns

Page step count

See Also

setPageStepCount(), singleStep()

12.132.2.16 void QwtWheel::paintEvent (QPaintEvent * event) [protected], [virtual]

Qt Paint Event.

Parameters

event | Paint event

12.132.2.17 void QwtWheel::setBorderWidth (int width)

Set the border width.

The border defaults to 2.

Parameters

width Border width

See Also

borderWidth()

12.132.2.18 void QwtWheel::setInverted (bool on)

En/Disable inverted appearance.

An inverted wheel increases its values in the opposite direction. The direction of an inverted horizontal wheel will be from right to left an inverted vertical wheel will increase from bottom to top.

Parameters

on En/Disable inverted appearance

See Also

isInverted()

12.132.2.19 void QwtWheel::setMass (double mass) [slot]

Set the slider's mass for flywheel effect.

If the slider's mass is greater then 0, it will continue to move after the mouse button has been released. Its speed decreases with time at a rate depending on the slider's mass. A large mass means that it will continue to move for a long time.

Derived widgets may overload this function to make it public.

Parameters

mass New mass in kg

See Also

mass()

12.132.2.20 void QwtWheel::setMaximum (double value)

Set the maximum value of the range

Parameters

value | Maximum value

See Also

setRange(), setMinimum(), maximum()

12.132.2.21 void QwtWheel::setMinimum (double value)

Set the minimum value of the range

| value | Minimum value |
|-------|---------------|
|-------|---------------|

See Also

setRange(), setMaximum(), minimum()

Note

The maximum is adjusted if necessary to ensure that the range remains valid.

12.132.2.22 void QwtWheel::setOrientation (Qt::Orientation orientation)

Set the wheel's orientation.

The default orientation is Qt::Horizontal.

Parameters

| orientation | Qt::Horizontal or Qt::Vertical. |
|-------------|---------------------------------|
|-------------|---------------------------------|

See Also

orientation()

12.132.2.23 void QwtWheel::setPageStepCount (int count)

Set the page step count.

pageStepCount is a multiplicator for the single step size that typically corresponds to the user pressing PageUp or PageDown.

A value of 0 disables page stepping.

The default value is 1.

Parameters

| count | Multiplicator for the single step size |
|-------|--|

See Also

pageStepCount(), setSingleStep()

12.132.2.24 void QwtWheel::setRange (double min, double max)

Set the minimum and maximum values.

The maximum is adjusted if necessary to ensure that the range remains valid. The value might be modified to be inside of the range.

Parameters

| min | Minimum value |
|-----|---------------|
| max | Maximum value |

See Also

minimum(), maximum()

12.132.2.25 void QwtWheel::setSingleStep (double stepSize)

Set the step size of the counter.

A value <= 0.0 disables stepping

Parameters

stepSize | Single step size

See Also

singleStep(), setPageStepCount()

12.132.2.26 void QwtWheel::setStepAlignment (bool on)

En/Disable step alignment.

When step alignment is enabled value changes initiated by user input (mouse, keyboard, wheel) are aligned to the multiples of the single step.

Parameters

| on On/Off | |
|-----------|--|
|-----------|--|

See Also

stepAlignment(), setSingleStep()

12.132.2.27 void QwtWheel::setTickCount (int count)

Adjust the number of grooves in the wheel's surface.

The number of grooves is limited to $6 \le \text{count} \le 50$. Values outside this range will be clipped. The default value is 10.

Parameters

| count | Number of grooves per 360 degrees |
|-------|-----------------------------------|

See Also

tickCount()

12.132.2.28 void QwtWheel::setTotalAngle (double angle) [slot]

Set the total angle which the wheel can be turned.

One full turn of the wheel corresponds to an angle of 360 degrees. A total angle of n*360 degrees means that the wheel has to be turned n times around its axis to get from the minimum value to the maximum value.

The default setting of the total angle is 360 degrees.

Parameters

| angle | total angle in degrees |
|-------|------------------------|

See Also

totalAngle()

12.132.2.29 void QwtWheel::setTracking (bool enable)

En/Disable tracking.

If tracking is enabled (the default), the wheel emits the valueChanged() signal while the wheel is moving. If tracking is disabled, the wheel emits the valueChanged() signal only when the wheel movement is terminated.

The wheelMoved() signal is emitted regardless id tracking is enabled or not.

| enable | On/Off |
|--------|--------|
| I I | |

See Also

isTracking()

12.132.2.30 void QwtWheel::setUpdateInterval (int interval)

Specify the update interval when the wheel is flying.

Default and minimum value is 50 ms.

Parameters

| interval | Interval in milliseconds |
|----------|--------------------------|

See Also

updateInterval(), setMass(), setTracking()

12.132.2.31 void QwtWheel::setValue (double value) [slot]

Set a new value without adjusting to the step raster.

Parameters

| value | New value |
|-------|-----------|
| | |

See Also

value(), valueChanged()

Warning

The value is clipped when it lies outside the range.

12.132.2.32 void QwtWheel::setViewAngle (double angle) [slot]

Specify the visible portion of the wheel.

You may use this function for fine-tuning the appearance of the wheel. The default value is 175 degrees. The value is limited from 10 to 175 degrees.

Parameters

| angle |
|-------|
|-------|

See Also

viewAngle(), setTotalAngle()

12.132.2.33 void QwtWheel::setWheelBorderWidth (int borderWidth)

Set the wheel border width of the wheel.

The wheel border must not be smaller than 1 and is limited in dependence on the wheel's size. Values outside the allowed range will be clipped.

The wheel border defaults to 2.

634 CONTENTS

Parameters

borderWidth Border width

See Also

internalBorder()

12.132.2.34 void QwtWheel::setWheelWidth (int width)

Set the width of the wheel.

Corresponds to the wheel height for horizontal orientation, and the wheel width for vertical orientation.

Parameters

width the wheel's width

See Also

wheelWidth()

12.132.2.35 void QwtWheel::setWrapping (bool on)

En/Disable wrapping.

If wrapping is true stepping up from maximum() value will take you to the minimum() value and vice versa.

Parameters

on En/Disable wrapping

See Also

wrapping()

12.132.2.36 double QwtWheel::singleStep () const

Returns

Single step size

See Also

setSingleStep()

12.132.2.37 QSize QwtWheel::sizeHint() const [protected], [virtual]

Returns

a size hint

12.132.2.38 bool QwtWheel::stepAlignment () const

Returns

True, when the step alignment is enabled

See Also

setStepAlignment(), singleStep()

```
12.132.2.39 int QwtWheel::tickCount ( ) const
Returns
      Number of grooves in the wheel's surface.
See Also
      setTickCnt()
12.132.2.40 void QwtWheel::timerEvent ( QTimerEvent * event ) [protected], [virtual]
Qt timer event.
The flying wheel effect is implemented using a timer
             event
                     Timer event
See Also
     updateInterval()
12.132.2.41 double QwtWheel::totalAngle ( ) const
Returns
      Total angle which the wheel can be turned.
See Also
      setTotalAngle()
12.132.2.42 int QwtWheel::updateInterval ( ) const
Returns
      Update interval when the wheel is flying
See Also
      setUpdateInterval(), mass(), isTracking()
12.132.2.43 double QwtWheel::value ( ) const
Returns
      Current value of the wheel
See Also
      setValue(), valueChanged()
12.132.2.44 double QwtWheel::valueAt (const QPoint & pos ) const [protected], [virtual]
Determine the value corresponding to a specified point
```

636 CONTENTS

Parameters

pos Position

Returns

Value corresponding to pos

12.132.2.45 void QwtWheel::valueChanged (double value) [signal]

Notify a change of value.

When tracking is enabled this signal will be emitted every time the value changes.

Parameters

value new value

See Also

setTracking()

12.132.2.46 double QwtWheel::viewAngle () const

Returns

Visible portion of the wheel

See Also

setViewAngle(), totalAngle()

12.132.2.47 int QwtWheel::wheelBorderWidth () const

Returns

Wheel border width

See Also

setWheelBorderWidth()

12.132.2.48 void QwtWheel::wheelEvent (QWheelEvent * event) [protected], [virtual]

Handle wheel events.

In/Decrement the value

Parameters

event Wheel event

12.132.2.49 void QwtWheel::wheelMoved (double value) [signal]

This signal is emitted when the user moves the wheel with the mouse.

Parameters

```
    value
    new value

    12.132.2.50
    void QwtWheel::wheelPressed ( ) [signal]

    This signal is emitted when the user presses the the wheel with the mouse

    12.132.2.51
    QRect QwtWheel::wheelRect ( ) const [protected]

    Returns
```

Rectangle of the wheel without the outer border

```
12.132.2.52 void QwtWheel::wheelReleased() [signal]
```

This signal is emitted when the user releases the mouse

```
12.132.2.53 int QwtWheel::wheelWidth ( ) const
```

Returns

Width of the wheel

See Also

setWheelWidth()

```
12.132.2.54 bool QwtWheel::wrapping ( ) const
```

Returns

True, when wrapping is set

See Also

setWrapping()

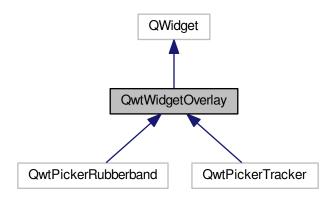
12.133 QwtWidgetOverlay Class Reference

An overlay for a widget.

```
#include <qwt_widget_overlay.h>
```

638 CONTENTS

Inheritance diagram for QwtWidgetOverlay:



Public Types

enum MaskMode { NoMask, MaskHint, AlphaMask }

Mask mode.

enum RenderMode { AutoRenderMode, CopyAlphaMask, DrawOverlay }

Render mode.

Public Member Functions

QwtWidgetOverlay (QWidget *)

Constructor.

virtual ~QwtWidgetOverlay ()

Destructor.

void setMaskMode (MaskMode)

Specify how to find the mask for the overlay.

- MaskMode maskMode () const
- void setRenderMode (RenderMode)
- RenderMode renderMode () const
- void updateOverlay ()
- virtual bool eventFilter (QObject *, QEvent *)

Event filter.

Protected Member Functions

- virtual void paintEvent (QPaintEvent *event)
- virtual void resizeEvent (QResizeEvent *event)
- virtual QRegion maskHint () const

Calculate an approximation for the mask.

virtual void drawOverlay (QPainter *painter) const =0

12.133.1 Detailed Description

An overlay for a widget.

The main use case of an widget overlay is to avoid heavy repaint operation of the widget below.

F.e. in combination with the plot canvas an overlay avoid replots as the content of the canvas can be restored from its backing store.

QwtWidgetOverlay is an abstract base class. Deriving classes are supposed to reimplement the following methods:

- drawOverlay()
- · maskHint()

Internally QwtPlotPicker uses overlays for displaying the rubber band and the tracker text.

See Also

QwtPlotCanvas::BackingStore

12.133.2 Member Enumeration Documentation

12.133.2.1 enum QwtWidgetOverlay::MaskMode

Mask mode.

When using masks the widget below gets paint events for the masked regions of the overlay only. Otherwise Qt triggers full repaints. On less powerful hardware (f.e embedded systems) - or when using the raster paint engine on a remote desktop - bit blitting is a noticeable operation, that needs to be avoided.

If and how to mask depends on how expensive the calculation of the mask is and how many pixels can be excluded by the mask.

The default setting is MaskHint.

See Also

setMaskMode(), maskMode()

Enumerator

NoMask Don't use a mask.

MaskHint Use maskHint() as mask. For many situations a fast approximation is good enough and it is not necessary to build a more detailed mask (f.e the bounding rectangle of a text).

AlphaMask Calculate a mask by checking the alpha values. Sometimes it is not possible to give a fast approximation and the mask needs to be calculated by drawing the overlay and testing the result.

When a valid maskHint() is available only pixels inside this approximation are checked.

12.133.2.2 enum QwtWidgetOverlay::RenderMode

Render mode.

For calculating the alpha mask the overlay has already been painted to a temporary Qlmage. Instead of rendering the overlay twice this buffer can be copied for drawing the overlay.

On graphic systems using the raster paint engine (QWS, Windows) it means usually copying some memory only. On X11 it results in an expensive operation building a pixmap and for simple overlays it might not be recommended.

640 CONTENTS

Note

The render mode has no effect, when maskMode() != AlphaMask.

Enumerator

AutoRenderMode Copy the buffer, when using the raster paint engine.

CopyAlphaMask Always copy the buffer.

DrawOverlay Never copy the buffer.

12.133.3 Constructor & Destructor Documentation

12.133.3.1 QwtWidgetOverlay::QwtWidgetOverlay (QWidget * widget)

Constructor.

Parameters

| widget | Parent widget, where the overlay is aligned to |
|--------|--|
|--------|--|

12.133.4 Member Function Documentation

12.133.4.1 virtual void QwtWidgetOverlay::drawOverlay (QPainter * *painter* **) const** [protected], [pure virtual]

Draw the widget overlay

Parameters

| painter | Painter |
|---------|---------|

12.133.4.2 bool QwtWidgetOverlay::eventFilter (QObject * object, QEvent * event) [virtual]

Event filter.

Resize the overlay according to the size of the parent widget.

Parameters

| object | Object to be filtered |
|--------|-----------------------|
| event | Event |

Returns

See QObject::eventFilter()

12.133.4.3 QRegion QwtWidgetOverlay::maskHint()const [protected],[virtual]

Calculate an approximation for the mask.

- · MaskHint The hint is used as mask.
- · AlphaMask The hint is used to speed up the algorithm for calculating a mask from non transparent pixels
- · NoMask The hint is unused.

The default implementation returns an invalid region indicating no hint.

Returns

Hint for the mask

12.133.4.4 QwtWidgetOverlay::MaskMode QwtWidgetOverlay::maskMode () const

Returns

Mode how to find the mask for the overlay

See Also

setMaskMode()

12.133.4.5 void QwtWidgetOverlay::paintEvent (QPaintEvent * event) [protected], [virtual]

Paint event

Parameters

event Paint event

See Also

drawOverlay()

12.133.4.6 QwtWidgetOverlay::RenderMode QwtWidgetOverlay::renderMode () const

Returns

Render mode

See Also

RenderMode, setRenderMode()

12.133.4.7 void QwtWidgetOverlay::resizeEvent (QResizeEvent * event) [protected], [virtual]

Resize event

Parameters

event Resize event

12.133.4.8 void QwtWidgetOverlay::setMaskMode (MaskMode mode)

Specify how to find the mask for the overlay.

Parameters

mode New mode

See Also

maskMode()

 $12.133.4.9 \quad \text{void QwtWidgetOverlay::setRenderMode (} \textbf{RenderMode} \textit{ mode })$

Set the render mode

642 CONTENTS

Parameters

mode Render mode

See Also

RenderMode, renderMode()

12.133.4.10 void QwtWidgetOverlay::updateOverlay ()

Recalculate the mask and repaint the overlay

Index

| ~QwtPlotDict | QwtWidgetOverlay, 639 |
|---------------------------|------------------------------|
| QwtPlotDict, 317 | AlwaysOff |
| \sim QwtScaleMap | QwtPicker, 237 |
| QwtScaleMap, 537 | AlwaysOn |
| | QwtPicker, 237 |
| abstractScaleDraw | append |
| QwtAbstractScale, 37 | QwtPicker, 240 |
| accept | QwtPlotPicker, 403 |
| QwtPicker, 239 | appended |
| QwtPlotZoomer, 480 | QwtPicker, 240 |
| activate | QwtPlotPicker, 403 |
| QwtPlotLayout, 365 | applyProperties |
| activated | QwtPlot, 268 |
| QwtPicker, 239 | Arrow |
| ActiveOnly | QwtDialSimpleNeedle, 127 |
| QwtPicker, 237 | arrowSize |
| addColorStop | QwtArrowButton, 70 |
| QwtLinearColorMap, 191 | aspectRatio |
| addItem | QwtPlotRescaler, 423 |
| QwtDynGridLayout, 130 | AtomicPainter |
| added | QwtPlotDirectPainter, 320 |
| QwtSetSample, 557 | attach |
| adjustedPoints | QwtPlotItem, 353 |
| QwtPicker, 239 | Attribute |
| alarmBrush | QwtPlotDirectPainter, 319 |
| QwtThermo, 610 | QwtScaleEngine, 531 |
| | - |
| alarmEnabled | attributes |
| QwtThermo, 610 | QwtScaleEngine, 532 |
| alarmLevel | Auto |
| QwtThermo, 610 | QwtSplineCurveFitter, 575 |
| alarmRect | AutoAdjustSamples |
| QwtThermo, 611 | QwtPlotAbstractBarChart, 286 |
| align | AutoCache |
| QwtLinearScaleEngine, 194 | QwtSymbol, 578 |
| QwtLogScaleEngine, 197 | AutoRenderMode |
| AlignScales | QwtWidgetOverlay, 640 |
| QwtPlotLayout, 365 | AutoScale |
| alignCanvasToScale | QwtPlotItem, 351 |
| QwtPlotLayout, 365 | AutoText |
| alignDate | QwtText, 595 |
| QwtDateScaleEngine, 111 | autoDelete |
| alignLegend | QwtPlotDict, 317 |
| QwtPlotLayout, 365 | autoReplot |
| alignScales | QwtPlot, 268 |
| QwtPlotLayout, 366 | autoScale |
| Alignment | QwtDateScaleEngine, 111 |
| QwtScaleDraw, 522 | QwtLinearScaleEngine, 194 |
| alignment | QwtLogScaleEngine, 197 |
| QwtKnob, 169 | QwtScaleEngine, 532 |
| QwtPlotLegendItem, 375 | Axis |
| QwtScaleDraw, 522 | QwtPlot, 267 |
| QwtScaleWidget, 542 | axisCnt |
| alpha | QwtPlot, 267 |
| QwtPlotRasterItem, 409 | axisAutoScale |
| AlphaMask | QwtPlot, 268 |
| • | • |

| axisEnabled | QwtPlotScaleItem, 431 |
|---|---|
| QwtPlot, 269 | BorderFlag |
| axisFont | QwtInterval, 153 |
| QwtPlot, 269 | borderFlags |
| axisInterval | QwtInterval, 154 |
| QwtPlot, 269 | borderPath |
| axisMaxMajor | QwtPlotCanvas, 299 |
| QwtPlot, 269 | QwtPlotGLCanvas, 324 |
| axisMaxMinor | borderPen |
| QwtPlot, 269 | QwtPlotLegendItem, 375 |
| axisScaleDiv | QwtText, 595 |
| QwtPlot, 270 | borderRadius |
| axisScaleDraw | QwtPlotCanvas, 299 |
| QwtPlot, 270 | QwtPlotLegendItem, 375 |
| axisScaleEngine | QwtText, 595 |
| QwtPlot, 270 | borderWidth |
| axisStepSize | |
| QwtPlot, 271 | QwtSlider, 564 |
| axisTitle | QwtThermo, 611 |
| QwtPlot, 271 | QwtWheel, 627 |
| axisValid | BottomLegend |
| QwtPlot, 271 | QwtPlot, 268 |
| | BottomScale |
| axisWidget | QwtScaleDraw, 522 |
| QwtPlot, 271 | BottomToTop |
| Backbone | QwtColumnRect, 76 |
| QwtAbstractScaleDraw, 44 | bounded |
| backgroundBrush | QwtLogTransform, 200 |
| QwtPlotLegendItem, 375 | QwtScaleDiv, 517 |
| QwtText, 595 | QwtTransform, 621 |
| BackgroundMode | boundingInterval |
| packoroungwooe | 9 |
| - | QwtOHLCSample, 219 |
| QwtPlotLegendItem, 374 | QwtOHLCSample, 219 |
| QwtPlotLegendItem, 374 backgroundMode | boundingLabelRect |
| QwtPlotLegendItem, 374 backgroundMode QwtPlotLegendItem, 375 | boundingLabelRect QwtScaleDraw, 523 |
| OwtPlotLegendItem, 374 backgroundMode OwtPlotLegendItem, 375 BackingStore | boundingLabelRect QwtScaleDraw, 523 boundingRect |
| QwtPlotLegendItem, 374 backgroundMode QwtPlotLegendItem, 375 BackingStore QwtPlotCanvas, 298 | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 |
| QwtPlotLegendItem, 374 backgroundMode QwtPlotLegendItem, 375 BackingStore QwtPlotCanvas, 298 backingStore | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 |
| QwtPlotLegendItem, 374 backgroundMode QwtPlotLegendItem, 375 BackingStore QwtPlotCanvas, 298 backingStore QwtPainter, 222 | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 |
| QwtPlotLegendItem, 374 backgroundMode QwtPlotLegendItem, 375 BackingStore QwtPlotCanvas, 298 backingStore QwtPainter, 222 QwtPlotCanvas, 299 | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 |
| OwtPlotLegendItem, 374 backgroundMode OwtPlotLegendItem, 375 BackingStore OwtPlotCanvas, 298 backingStore OwtPainter, 222 OwtPlotCanvas, 299 Bar | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 |
| OwtPlotLegendItem, 374 backgroundMode OwtPlotLegendItem, 375 BackingStore OwtPlotCanvas, 298 backingStore OwtPainter, 222 OwtPlotCanvas, 299 Bar OwtIntervalSymbol, 164 | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 |
| OwtPlotLegendItem, 374 backgroundMode OwtPlotLegendItem, 375 BackingStore OwtPlotCanvas, 298 backingStore OwtPainter, 222 OwtPlotCanvas, 299 Bar | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 |
| OwtPlotLegendItem, 374 backgroundMode OwtPlotLegendItem, 375 BackingStore OwtPlotCanvas, 298 backingStore OwtPainter, 222 OwtPlotCanvas, 299 Bar OwtIntervalSymbol, 164 | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 QwtPlotItem, 353 |
| QwtPlotLegendItem, 374 backgroundMode QwtPlotLegendItem, 375 BackingStore QwtPlotCanvas, 298 backingStore QwtPainter, 222 QwtPlotCanvas, 299 Bar QwtIntervalSymbol, 164 QwtPlotTradingCurve, 466 | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 QwtPlotItem, 353 QwtPlotMarker, 386 |
| QwtPlotLegendItem, 374 backgroundMode QwtPlotLegendItem, 375 BackingStore QwtPlotCanvas, 298 backingStore QwtPainter, 222 QwtPlotCanvas, 299 Bar QwtIntervalSymbol, 164 QwtPlotTradingCurve, 466 barTitle | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 QwtPlotItem, 353 QwtPlotMarker, 386 QwtPlotMultiBarChart, 392 |
| OwtPlotLegendItem, 374 backgroundMode | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 QwtPlotItem, 353 QwtPlotMarker, 386 QwtPlotMultiBarChart, 392 QwtPlotRasterItem, 409 |
| OwtPlotLegendItem, 374 backgroundMode OwtPlotLegendItem, 375 BackingStore OwtPlotCanvas, 298 backingStore OwtPainter, 222 OwtPlotCanvas, 299 Bar OwtIntervalSymbol, 164 OwtPlotTradingCurve, 466 barTitle OwtPlotBarChart, 292 barTitles | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 QwtPlotIdem, 353 QwtPlotMarker, 386 QwtPlotMultiBarChart, 392 QwtPlotRasterItem, 409 QwtPlotSeriesItem, 436 |
| OwtPlotLegendItem, 374 backgroundMode | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 QwtPlotItem, 353 QwtPlotMarker, 386 QwtPlotMultiBarChart, 392 QwtPlotRasterItem, 409 |
| OwtPlotLegendItem, 374 backgroundMode | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 QwtPlotIdem, 353 QwtPlotMarker, 386 QwtPlotMultiBarChart, 392 QwtPlotRasterItem, 409 QwtPlotSeriesItem, 436 |
| OwtPlotLegendItem, 374 backgroundMode | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 QwtPlotIdem, 353 QwtPlotMarker, 386 QwtPlotMultiBarChart, 392 QwtPlotRasterItem, 409 QwtPlotSeriesItem, 436 QwtPlotTradingCurve, 467 |
| OwtPlotLegendItem, 374 backgroundMode | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 QwtPlotMarker, 386 QwtPlotMultiBarChart, 392 QwtPlotRasterItem, 409 QwtPlotSeriesItem, 436 QwtPlotTradingCurve, 467 QwtPlotZoneItem, 475 |
| OwtPlotLegendItem, 374 backgroundMode OwtPlotLegendItem, 375 BackingStore OwtPlotCanvas, 298 backingStore OwtPainter, 222 OwtPlotCanvas, 299 Bar OwtIntervalSymbol, 164 OwtPlotTradingCurve, 466 barTitle OwtPlotBarChart, 292 barTitles OwtPlotMultiBarChart, 392 base OwtScaleEngine, 532 baseline OwtPlotAbstractBarChart, 286 OwtPlotCurve, 307 | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 QwtPlotIdem, 353 QwtPlotMarker, 386 QwtPlotMultiBarChart, 392 QwtPlotRasterItem, 409 QwtPlotSeriesItem, 436 QwtPlotZoneltem, 475 QwtPoint3DSeriesData, 489 |
| OwtPlotLegendItem, 374 backgroundMode | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 QwtPlotIdem, 353 QwtPlotMarker, 386 QwtPlotMultiBarChart, 392 QwtPlotRasterItem, 409 QwtPlotSeriesItem, 436 QwtPlotZoneItem, 475 QwtPoint3DSeriesData, 489 QwtPointMapper, 492 |
| OwtPlotLegendItem, 374 backgroundMode | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 QwtPlotItem, 353 QwtPlotMarker, 386 QwtPlotMultiBarChart, 392 QwtPlotRasterItem, 409 QwtPlotSeriesItem, 436 QwtPlotZoneItem, 475 QwtPoint3DSeriesData, 489 QwtPointArrayData, 490 |
| OwtPlotLegendItem, 374 backgroundMode | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 QwtPlotIntervalCurve, 343 QwtPlotMarker, 386 QwtPlotMultiBarChart, 392 QwtPlotRasterItem, 409 QwtPlotSeriesItem, 436 QwtPlotZoneItem, 475 QwtPoint3DSeriesData, 489 QwtPointMapper, 492 QwtPointSeriesData, 500 QwtSeriesData, 552 |
| OwtPlotLegendItem, 374 backgroundMode | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 QwtPlotMarker, 386 QwtPlotMultiBarChart, 392 QwtPlotRasterItem, 409 QwtPlotSeriesItem, 436 QwtPlotTradingCurve, 467 QwtPlotZoneItem, 475 QwtPoint3DSeriesData, 489 QwtPointMapper, 492 QwtPointSeriesData, 500 QwtSeriesData, 552 QwtSetSeriesData, 559 |
| OwtPlotLegendItem, 374 backgroundMode | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 QwtPlotMarker, 386 QwtPlotMultiBarChart, 392 QwtPlotRasterItem, 409 QwtPlotSeriesItem, 436 QwtPlotTradingCurve, 467 QwtPlotZoneItem, 475 QwtPoint3DSeriesData, 489 QwtPointArrayData, 490 QwtPointMapper, 492 QwtPointSeriesData, 500 QwtSeriesData, 552 QwtSetSeriesData, 559 QwtSymbol, 580 |
| OwtPlotLegendItem, 374 backgroundMode | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 QwtPlotItem, 353 QwtPlotMarker, 386 QwtPlotMultiBarChart, 392 QwtPlotRasterItem, 409 QwtPlotSeriesItem, 436 QwtPlotZoneItem, 475 QwtPoint3DSeriesData, 489 QwtPointArrayData, 490 QwtPointMapper, 492 QwtPointSeriesData, 552 QwtSeriesData, 559 QwtSymbol, 580 QwtSyntheticPointData, 589 |
| OwtPlotLegendItem, 374 backgroundMode | boundingLabelRect QwtScaleDraw, 523 boundingRect QwtCPointerData, 98 QwtDial, 117 QwtGraphic, 144 QwtIntervalSeriesData, 163 QwtPlotBarChart, 292 QwtPlotHistogram, 335 QwtPlotHistogram, 335 QwtPlotIntervalCurve, 343 QwtPlotMarker, 386 QwtPlotMultiBarChart, 392 QwtPlotRasterItem, 409 QwtPlotSeriesItem, 436 QwtPlotTradingCurve, 467 QwtPlotZoneItem, 475 QwtPoint3DSeriesData, 489 QwtPointArrayData, 490 QwtPointMapper, 492 QwtPointSeriesData, 500 QwtSeriesData, 552 QwtSetSeriesData, 559 QwtSymbol, 580 |

| QwtColumnSymbol, 77 | QwtPlotLayout, 366 |
|---------------------------|----------------------------|
| QwtIntervalSymbol, 164 | canvasRect |
| brush | QwtPlotLayout, 366 |
| QwtIntervalSymbol, 164 | canvasResizeEvent |
| QwtPlotCurve, 307 | QwtPlotRescaler, 423 |
| QwtPlotHistogram, 335 | ceil |
| QwtPlotIntervalCurve, 343 | QwtDate, 101 |
| QwtPlotShapeItem, 440 | ceilEps |
| QwtPlotZoneItem, 475 | QwtScaleArithmetic, 513 |
| QwtSymbol, 580 | changeEvent |
| buildInterval | QwtDial, 118 |
| QwtScaleEngine, 532 | QwtKnob, 170 |
| buildMajorTicks | QwtSlider, 565 |
| QwtLinearScaleEngine, 195 | QwtThermo, 611 |
| QwtLogScaleEngine, 198 | changed |
| buildMinorTicks | QwtPicker, 240 |
| QwtLinearScaleEngine, 195 | ChartStyle |
| QwtLogScaleEngine, 198 | QwtPlotMultiBarChart, 392 |
| buildNaturalSpline | Checkable |
| QwtSpline, 572 | QwtLegendData, 183 |
| buildPeriodicSpline | checked |
| QwtSpline, 572 | QwtLegend, 177 |
| buildTicks | chunkSize |
| QwtLinearScaleEngine, 195 | QwtWeedingCurveFitter, 623 |
| QwtLogScaleEngine, 198 | Clickable |
| Button | QwtLegendData, 183 |
| QwtCounter, 90 | clicked |
| Button1 | QwtLegend, 178 |
| QwtCounter, 90 | ClipPoints |
| Button2 | QwtPlotSpectroCurve, 445 |
| QwtCounter, 90 | ClipPolygons |
| Button3 | QwtPlotCurve, 305 |
| QwtCounter, 91 | QwtPlotIntervalCurve, 343 |
| ButtonCnt | QwtPlotShapeltem, 439 |
| QwtCounter, 91 | ClipSymbol |
| buttonReleased | QwtPlotIntervalCurve, 343 |
| QwtCounter, 91 | ClipSymbols |
| | QwtPlotTradingCurve, 466 |
| Cache | clipCircle |
| QwtSymbol, 578 | QwtClipper, 71 |
| CachePolicy | clipPolygon |
| QwtPlotRasterItem, 408 | QwtClipper, 72 |
| QwtSymbol, 578 | clipPolygonF |
| cachePolicy | QwtClipper, 72 |
| QwtPlotRasterItem, 409 | clipRegion |
| QwtSymbol, 580 | |
| CandleStick | QwtPlotDirectPainter, 320 |
| QwtPlotTradingCurve, 466 | closePolyline |
| canvas | QwtPlotCurve, 307 |
| QwtPlot, 272 | closestPoint |
| QwtPlotPicker, 403 | QwtPlotCurve, 307 |
| QwtPlotRescaler, 423 | coefficientsA |
| CanvasFocusIndicator | QwtSpline, 572 |
| QwtPlotCanvas, 298 | coefficientsB |
| canvasBackground | QwtSpline, 572 |
| QwtPlot, 272 | coefficientsC |
| canvasMap | QwtSpline, 572 |
| QwtPlot, 272 | color |
| canvasMargin | QwtAlphaColorMap, 62 |
| | |

| QwtColorMap, 74 | QwtNullTransform, 218 |
|---|------------------------------|
| color1 | QwtPowerTransform, 502 |
| QwtLinearColorMap, 191 | CopyAlphaMask |
| color2 | QwtWidgetOverlay, 640 |
| | CopyBackingStore |
| QwtLinearColorMap, 191 colorBarInterval | QwtPlotDirectPainter, 320 |
| | |
| QwtScaleWidget, 543 | Count |
| colorBarRect | QwtDynGridLayout, 130 |
| QwtScaleWidget, 543 | createWidget |
| colorBarWidth | QwtLegend, 178 |
| QwtScaleWidget, 543 | Cross |
| colorIndex | QwtPlotMarker, 385 |
| QwtColorMap, 74 | QwtSymbol, 579 |
| QwtLinearColorMap, 192 | CrossRubberBand |
| colorMap | QwtPicker, 238 |
| QwtPlotSpectroCurve, 445 | cursor |
| QwtPlotSpectrogram, 450 | QwtPanner, 230 |
| QwtScaleWidget, 543 | CurveAttribute |
| QwtThermo, 611 | QwtPlotCurve, 304 |
| colorRange | curveFitter |
| QwtPlotSpectroCurve, 445 | QwtPlotCurve, 308 |
| colorStops | CurveStyle |
| QwtLinearColorMap, 192 | QwtPlotCurve, 304 |
| colorTable | QwtPlotIntervalCurve, 342 |
| | , |
| QwtColorMap, 74 | DTriangle |
| columnRect | QwtSymbol, 579 |
| QwtPlotHistogram, 335 | data |
| Columns | QwtLegendLabel, 188 |
| QwtPlotHistogram, 334 | QwtPlotSpectrogram, 451, 452 |
| columnsForWidth | QwtSeriesStore, 553 |
| QwtDynGridLayout, 130 | dataRect |
| commands | QwtAbstractSeriesStore, 51 |
| QwtGraphic, 144 | QwtSeriesStore, 554 |
| ConrecFlag | dataSize |
| QwtRasterData, 503 | QwtAbstractSeriesStore, 51 |
| contains | QwtSeriesStore, 554 |
| QwtInterval, 154 | dateFormat |
| QwtScaleDiv, 518 | QwtDateScaleDraw, 107 |
| QwtScaleEngine, 532 | dateFormatOfDate |
| contentsMask | QwtDateScaleDraw, 107 |
| QwtPanner, 230 | dateOfWeek0 |
| QwtPlotPanner, 399 | QwtDate, 101 |
| contentsWidget | |
| QwtLegend, 178 | Day QwtDate, 101 |
| ContourMode | |
| QwtPlotSpectrogram, 450 | Decreasing |
| contourLevels | QwtPlotTradingCurve, 466 |
| | DefaultLayout |
| QwtPlotSpectrogram, 450 | QwtPlotRenderer, 414 |
| contourLines | defaultContourPen |
| QwtRasterData, 503 | QwtPlotSpectrogram, 452 |
| contourPen | defaultIcon |
| QwtPlotSpectrogram, 451 | QwtPlotItem, 353 |
| contourRasterSize | defaultItemMode |
| QwtPlotSpectrogram, 451 | QwtLegend, 179 |
| controlPointRect | defaultSize |
| QwtGraphic, 144 | QwtGraphic, 144 |
| сору | detach |
| QwtLogTransform, 200 | QwtPlotItem, 354 |

| | 0 10 10 11 10 |
|---------------------------|---------------------------|
| detachltems | QwtPlotRasterItem, 409 |
| QwtPlotDict, 317 | QwtPlotSeriesItem, 436 |
| Diamond | QwtPlotShapeItem, 440 |
| QwtSymbol, 579 | QwtPlotSpectrogram, 452 |
| dimForLength | QwtPlotSvgltem, 458 |
| QwtScaleWidget, 543 | QwtPlotTextLabel, 461 |
| Direction | QwtPlotZoneItem, 475 |
| QwtColumnRect, 76 | QwtRichTextEngine, 506 |
| QwtPlotTradingCurve, 466 | QwtSimpleCompassRose, 560 |
| DiscardBackground | QwtText, 596 |
| - | |
| QwtPlotRenderer, 413 | QwtTextEngine, 602 |
| DiscardCanvasBackground | DrawOverlay |
| QwtPlotRenderer, 413 | QwtWidgetOverlay, 640 |
| DiscardCanvasFrame | drawArrow |
| QwtPlotRenderer, 413 | QwtArrowButton, 70 |
| DiscardFooter | drawBackbone |
| QwtPlotRenderer, 413 | QwtAbstractScaleDraw, 44 |
| DiscardLegend | QwtRoundScaleDraw, 509 |
| QwtPlotRenderer, 413 | QwtScaleDraw, 523 |
| DiscardNone | drawBackgound |
| QwtPlotRenderer, 413 | QwtPainter, 222 |
| DiscardTitle | drawBackground |
| | <u> </u> |
| QwtPlotRenderer, 413 | QwtPlotGLCanvas, 324 |
| DiscardFlag | QwtPlotLegendItem, 376 |
| QwtPlotRenderer, 413 | drawBar |
| discardFlags | QwtPlotBarChart, 292 |
| QwtPlotRenderer, 414 | QwtPlotMultiBarChart, 392 |
| discardRaster | QwtPlotTradingCurve, 467 |
| QwtRasterData, 504 | drawBorder |
| DisplayMode | QwtPlotCanvas, 299 |
| QwtPicker, 237 | QwtPlotGLCanvas, 324 |
| QwtPlotSpectrogram, 450 | drawBox |
| divideEps | QwtColumnSymbol, 78 |
| QwtScaleArithmetic, 513 | drawButtonLabel |
| divideInterval | QwtArrowButton, 70 |
| QwtScaleArithmetic, 514 | drawCandleStick |
| | |
| QwtScaleEngine, 533 | QwtPlotTradingCurve, 467 |
| divideScale | drawCanvas |
| QwtDateScaleEngine, 112 | QwtPlot, 272 |
| QwtLinearScaleEngine, 195 | drawColorBar |
| QwtLogScaleEngine, 198 | QwtPainter, 222 |
| QwtScaleEngine, 533 | QwtScaleWidget, 543 |
| Dot | drawColumn |
| QwtKnob, 169 | QwtPlotHistogram, 335 |
| Dots | drawColumns |
| QwtPlotCurve, 305 | QwtPlotHistogram, 337 |
| draw | drawContents |
| QwtAbstractScaleDraw, 44 | QwtDial, 118 |
| QwtColumnSymbol, 77 | drawContourLines |
| QwtCompassRose, 84 | QwtPlotSpectrogram, 452 |
| QwtDialNeedle, 125 | drawCurve |
| | |
| QwtIntervalSymbol, 164 | QwtPlotCurve, 308 |
| QwtMathMLTextEngine, 209 | drawDots |
| QwtPlainTextEngine, 262 | QwtPlotCurve, 308 |
| QwtPlotGrid, 328 | QwtPlotSpectroCurve, 446 |
| QwtPlotItem, 354 | drawFocusIndicator |
| QwtPlotLegendItem, 376 | QwtDial, 118 |
| QwtPlotMarker, 386 | QwtKnob, 170 |
| | |

| QwtPlotCanvas, 299 | QwtPlotMultiBarChart, 393 |
|--------------------------------------|---------------------------|
| drawFrame | drawScale |
| QwtDial, 118 | QwtDial, 119 |
| QwtPainter, 222 | drawScaleContents |
| drawGroupedBars | QwtCompass, 81 |
| QwtPlotMultiBarChart, 393 | QwtDial, 119 |
| drawHand | drawSeries |
| QwtAnalogClock, 66 | QwtPlotBarChart, 293 |
| drawHandle | QwtPlotCurve, 309 |
| QwtSlider, 565 | QwtPlotDirectPainter, 320 |
| drawlmage | QwtPlotHistogram, 338 |
| QwtGraphic, 144 | QwtPlotIntervalCurve, 344 |
| drawltems | QwtPlotMultiBarChart, 393 |
| QwtPlot, 272 | QwtPlotSeriesItem, 436 |
| QwtPlotGLCanvas, 324 | QwtPlotSpectroCurve, 446 |
| drawKnob | QwtPlotTradingCurve, 467 |
| QwtKnob, 170 | drawSimpleRichText |
| drawLabel | QwtPainter, 223 |
| QwtAbstractScaleDraw, 44 | drawSlider |
| QwtPlotMarker, 386 | QwtSlider, 565 |
| QwtRoundScaleDraw, 509 | drawStackedBars |
| QwtScaleDraw, 523 | QwtPlotMultiBarChart, 394 |
| drawLegendData | drawSteps |
| QwtPlotLegendItem, 376 | QwtPlotCurve, 309 |
| drawLines | drawSticks |
| QwtPlotCurve, 308 | QwtPlotCurve, 309 |
| QwtPlotHistogram, 337 | drawSymbol |
| QwtPlotMarker, 386 | QwtSymbol, 581 |
| drawLiquid | drawSymbols |
| QwtThermo, 611 | QwtPlotCurve, 310 |
| drawMarker | QwtPlotIntervalCurve, 344 |
| | QwtPlotTradingCurve, 468 |
| QwtKnob, 170 drawNeedle | QwtSymbol, 581 |
| | drawTick |
| QwtAnalogClock, 66 | QwtAbstractScaleDraw, 44 |
| QwtCompassMagnetNeedle, 83 | QwtRoundScaleDraw, 509 |
| QwtCompassWindArrow, 88 | QwtScaleDraw, 523 |
| QwtDial, 118 | drawTicks |
| QwtDialNeedle, 125 | QwtWheel, 627 |
| QwtDialSimpleNeedle, 127 drawOutline | drawTitle |
| | QwtScaleWidget, 544 |
| QwtPlotHistogram, 337 | drawTracker |
| drawOverlay | QwtPicker, 241 |
| QwtWidgetOverlay, 640 | drawTube |
| drawPath | QwtPlotIntervalCurve, 344 |
| QwtGraphic, 146 | drawUserSymbol |
| drawPixmap | QwtPlotTradingCurve, 468 |
| QwtGraphic, 146 | drawWheelBackground |
| drawRose | QwtWheel, 627 |
| QwtCompass, 81 | |
| QwtSimpleCompassRose, 560 | elapsed |
| drawRoundFrame | QwtSamplingThread, 512 |
| QwtPainter, 223 | QwtSystemClock, 592 |
| drawRoundedFrame | Ellipse |
| QwtPainter, 223 | QwtSymbol, 579 |
| drawRubberBand | EllipseRubberBand |
| QwtPicker, 240 | QwtPicker, 238 |
| drawSample | enableAxis |
| QwtPlotBarChart, 293 | QwtPlot, 274 |

| enableComponent | extent |
|--|------------------------------|
| QwtAbstractScaleDraw, 46 | QwtAbstractScaleDraw, 46 |
| enableX | QwtRoundScaleDraw, 509 |
| QwtPlotGrid, 328 | QwtScaleDraw, 524 |
| enableXMin | GWIOGGIODIAW, OZ I |
| QwtPlotGrid, 328 | fillBrush |
| , and the second | QwtThermo, 613 |
| enableY | fillCurve |
| QwtPlotGrid, 329 | QwtPlotCurve, 310 |
| enableYMin | fillPixmap |
| QwtPlotGrid, 329 | QwtPainter, 223 |
| end | fillRect |
| QwtPicker, 241 | QwtThermo, 613 |
| QwtPlotPicker, 403 | FilterPoints |
| QwtPlotZoomer, 481 | QwtPlotCurve, 305 |
| endBorderDist | FirstDay |
| QwtScaleWidget, 544 | QwtDate, 101 |
| event | FirstThursday |
| QwtCounter, 91 | QwtDate, 101 |
| QwtPlot, 274 | |
| QwtPlotCanvas, 300 | fitCurve |
| QwtPlotGLCanvas, 324 | QwtCurveFitter, 99 |
| eventFilter | QwtSplineCurveFitter, 575 |
| QwtLegend, 179 | QwtWeedingCurveFitter, 623 |
| QwtMagnifier, 202 | FitMode |
| QwtPanner, 231 | QwtSplineCurveFitter, 575 |
| QwtPicker, 241 | fitMode |
| QwtPlot, 274 | QwtSplineCurveFitter, 575 |
| QwtWidgetOverlay, 640 | Fitted |
| ExcludeBorders | QwtPlotCurve, 304 |
| QwtInterval, 153 | Fitting |
| | QwtPlotRescaler, 421 |
| ExcludeMaximum Outleton of 152 | Fixed |
| QwtInterval, 153 | QwtPlotRescaler, 421 |
| ExcludeMinimum | FixedColors |
| QwtInterval, 153 | QwtLinearColorMap, 191 |
| ExpandBoth | FixedSampleSize |
| QwtPlotRescaler, 421 | QwtPlotAbstractBarChart, 286 |
| ExpandDown | flags |
| QwtPlotRescaler, 421 | QwtPointMapper, 492 |
| ExpandUp | Flat |
| QwtPlotRescaler, 421 | QwtKnob, 169 |
| expandInterval | Floating |
| QwtPlotRescaler, 424 | QwtScaleEngine, 531 |
| expandLineBreaks | floor |
| QwtPlotLayout, 366 | QwtDate, 102 |
| expandScale | floorEps |
| QwtPlotRescaler, 424 | QwtScaleArithmetic, 514 |
| Expanding | FocusIndicator |
| QwtPlotRescaler, 421 | QwtPlotCanvas, 297 |
| ExpandingDirection | focusIndicator |
| QwtPlotRescaler, 421 | QwtPlotCanvas, 300 |
| expandingDirection | font |
| QwtPlotRescaler, 423 | QwtPlotLegendItem, 376 |
| expandingDirections | QwtPlotScaleItem, 431 |
| QwtDynGridLayout, 130 | footer |
| exportTo | |
| QwtPlotRenderer, 414 | QwtPlot, 274 footerLabel |
| | |
| extend Outlintown 154 | QwtPlot, 275 |
| QwtInterval, 154 | footerRect |

| 0.151.11 | |
|--|----------------------------------|
| QwtPlotLayout, 367 | HackStyledBackground |
| Format | QwtPlotCanvas, 298 |
| QwtColorMap, 73 | Hand |
| format | QwtAnalogClock, 64 |
| QwtColorMap, 74 | hand |
| FrameWithScales | QwtAnalogClock, 66 handleRect |
| QwtPlotRenderer, 414 frameRect | |
| QwtPlotGLCanvas, 324 | QwtSlider, 565 handleSize |
| frameShadow | |
| QwtDial, 119 | QwtSlider, 565 hasClipping |
| QwtPlotGLCanvas, 324 | QwtPlotDirectPainter, 320 |
| frameShape | hasComponent |
| QwtPlotGLCanvas, 325 | QwtAbstractScaleDraw, 46 |
| FrameStyle | hasGroove |
| QwtColumnSymbol, 77 | QwtSlider, 565 |
| frameStyle | hasHeightForWidth |
| QwtColumnSymbol, 78 | QwtDynGridLayout, 130 |
| QwtPlotGLCanvas, 325 | hasRole |
| frameWidth | QwtLegendData, 184 |
| QwtPlotGLCanvas, 325 | hasTrough |
| FullRepaint | QwtSlider, 565 |
| QwtPlotDirectPainter, 320 | heightForWidth |
| | QwtDynGridLayout, 130 |
| geometry | QwtLegend, 179 |
| QwtPlotLegendItem, 376 | QwtMathMLTextEngine, 209 |
| getBorderDistHint | QwtPlainTextEngine, 263 |
| QwtScaleDraw, 524 | QwtPlotLegendItem, 377 |
| QwtScaleWidget, 544 | QwtRichTextEngine, 506 |
| getCanvasMarginHint | QwtText, 596 |
| QwtPlotAbstractBarChart, 287 | QwtTextEngine, 602 |
| QwtPlotItem, 354 | QwtTextLabel, 605 |
| getCanvasMarginsHint | Hexagon |
| QwtPlot, 275 | QwtSymbol, 579 |
| getMinBorderDist | HistogramStyle |
| QwtScaleWidget, 544 | QwtPlotHistogram, 334 |
| getMouseButton | horizontalScrollBar |
| QwtMagnifier, 202 | QwtLegend, 179 |
| getZoomInKey | Hour |
| QwtMagnifier, 202 | QwtDate, 101 |
| getZoomOutKey QwtMagnifier, 203 | HourHand |
| • | QwtAnalogClock, 64 |
| grab QwtPanner, 231 | icon |
| QwtPlotPanner, 399 | QwtLegendData, 184 |
| grabProperties | QwtLegendLabel, 188 |
| QwtPlot, 275 | IgnoreAllVerticesOnLevel |
| Graphic | QwtRasterData, 503 |
| QwtSymbol, 579 | IgnoreFooter |
| graphic | QwtPlotLayout, 365 |
| QwtSymbol, 581 | IgnoreFrames |
| Grouped | QwtPlotLayout, 365 |
| QwtPlotMultiBarChart, 392 | IgnoreLegend |
| and constant of the constant o | QwtPlotLayout, 365 |
| HLine | IgnoreOutOfRange |
| QwtPlotMarker, 385 | QwtRasterData, 503 |
| QwtSymbol, 579 | IgnoreScrollbars |
| HLineRubberBand | QwtPlotLayout, 365 |
| QwtPicker, 238 | IgnoreTitle |
| | - |

| QwtPlotLayout, 365 | IntervalType |
|----------------------------|---------------------------------|
| Image | QwtDate, 100 |
| QwtPainterCommand, 226 | intervalType |
| ImageBuffer | QwtDateScaleDraw, 107 |
| QwtPlotCurve, 305 | QwtDateScaleEngine, 112 |
| ImageMode | invTransform |
| QwtPlotSpectrogram, 450 | QwtAbstractScale, 37 |
| imageData | QwtLogTransform, 200 |
| QwtPainterCommand, 227 | QwtNullTransform, 218 |
| imageMap | QwtPlot, 276 |
| QwtPlotRasterItem, 409 | QwtPlotPicker, 404 |
| ImmediatePaint | QwtPowerTransform, 502 |
| QwtPlotCanvas, 298 | QwtScaleMap, 537 |
| incSteps | QwtTransform, 621 |
| QwtCounter, 91 | Invalid |
| IncludeBorders | QwtPainterCommand, 226 |
| QwtInterval, 153 | invalidate |
| IncludeReference | QwtInterval, 155 |
| QwtScaleEngine, 531 | QwtPlotLayout, 367 |
| Increasing | invalidateCache |
| QwtPlotTradingCurve, 466 | QwtAbstractScaleDraw, 46 |
| incrementValue | QwtDial, 119 |
| QwtAbstractSlider, 54 | QwtPlotRasterItem, 410 |
| incrementedValue | QwtSymbol, 582 |
| QwtAbstractSlider, 54 | invert |
| index | QwtScaleDiv, 518 |
| QwtPixelMatrix, 260 | Inverted |
| Indexed | QwtPlotCurve, 304 |
| QwtColorMap, 74 | QwtScaleEngine, 531 |
| infoToItem | inverted |
| QwtPlot, 275 | QwtInterval, 155 |
| initKeyPattern | QwtScaleDiv, 518 |
| QwtEventPattern, 137 | invertedControls |
| initMousePattern | QwtAbstractSlider, 54 |
| QwtEventPattern, 137 | isActive |
| initRaster | QwtPicker, 241 |
| QwtRasterData, 504 | isAligning |
| innerRect | QwtPainter, 224 |
| QwtDial, 119 | isAxisEnabled |
| insertItem | QwtPlotMagnifier, 383 |
| QwtPlotDict, 317 | QwtPlotPanner, 400 |
| insertLegend | isColorBarEnabled |
| QwtPlot, 276 | QwtScaleWidget, 545 |
| intersect | isEmpty |
| QwtInterval, 154 | QwtAbstractLegend, 33 |
| intersects | QwtDynGridLayout, 131 |
| QwtInterval, 155 | QwtGraphic, 146 |
| interval | QwtLegend, 179 |
| QwtPlotRasterItem, 410 | QwtText, 596 |
| QwtPlotRescaler, 424 | isEnabled |
| QwtPlotSpectrogram, 453 | QwtMagnifier, 203 |
| QwtPlotZoneItem, 475 | QwtPanner, 231 |
| QwtRasterData, 504 | QwtPicker, 242 |
| | |
| QwtSamplingThread, 512 | QwtPlotRescaler, 424 isInverted |
| QwtScaleDiv, 518 | |
| QwtSyntheticPointData, 589 | QwtAbstractScale, 37 |
| intervalHint | QwtWheel, 627 |
| QwtPlotRescaler, 424 | isInverting |
| | |

| QwtScaleMap, 538 | itemMargin |
|------------------------|------------------------|
| isNull | QwtPlotLegendItem, 377 |
| QwtGraphic, 146 | itemMode |
| QwtInterval, 155 | QwtLegendLabel, 188 |
| QwtPoint3D, 487 | itemSpacing |
| QwtSystemClock, 592 | QwtPlotLegendItem, 377 |
| QwtText, 596 | itemToInfo |
| isOrientationEnabled | QwtPlot, 277 |
| QwtPanner, 231 | Will lot, 277 |
| isPinPointEnabled | JulianDayForEpoch |
| | QwtDate, 100 |
| QwtSymbol, 582 | |
| isReadOnly | KeepSize |
| QwtAbstractSlider, 55 | QwtPicker, 238 |
| QwtCounter, 91 | KeyAbort |
| isScaleDivFromAxis | QwtEventPattern, 136 |
| QwtPlotScaleItem, 431 | KeyDown |
| isScrollPosition | QwtEventPattern, 136 |
| QwtAbstractSlider, 55 | KeyHome |
| QwtDial, 119 | QwtEventPattern, 136 |
| QwtKnob, 170 | KeyLeft |
| QwtSlider, 566 | QwtEventPattern, 136 |
| isTracking | KeyPatternCount |
| QwtAbstractSlider, 55 | QwtEventPattern, 136 |
| QwtWheel, 627 | |
| isValid | KeyRedo |
| QwtAbstractSlider, 55 | QwtEventPattern, 136 |
| QwtCounter, 92 | KeyRight |
| QwtInterval, 155 | QwtEventPattern, 136 |
| QwtLegendData, 184 | KeySelect1 |
| QwtOHLCSample, 219 | QwtEventPattern, 136 |
| isVisible | KeySelect2 |
| QwtPlotItem, 354 | QwtEventPattern, 136 |
| isX11GraphicsSystem | KeyUndo |
| QwtPainter, 224 | QwtEventPattern, 136 |
| ItemBackground | KeyUp |
| QwtPlotLegendItem, 375 | QwtEventPattern, 136 |
| | keyFactor |
| ItemFocusIndicator | QwtMagnifier, 203 |
| QwtPlotCanvas, 298 | keyMatch |
| itemAt | QwtEventPattern, 137 |
| QwtDynGridLayout, 131 | keyPattern |
| itemAttached | QwtEventPattern, 138 |
| QwtPlot, 276 | KeyPatternCode |
| ItemAttribute | QwtEventPattern, 135 |
| QwtPlotItem, 351 | keyPressEvent |
| itemChanged | QwtAbstractSlider, 55 |
| QwtPlotItem, 355 | QwtCompass, 81 |
| itemChecked | QwtCounter, 92 |
| QwtLegend, 179 | QwtWheel, 627 |
| itemClicked | knobRect |
| QwtLegend, 180 | QwtKnob, 170 |
| itemCount | KnobStyle |
| QwtDynGridLayout, 131 | QwtKnob, 169 |
| itemInfo | knobStyle |
| QwtLegend, 180 | QwtKnob, 171 |
| ItemInterest | QWILLIOD, 171 |
| QwtPlotItem, 352 | LTriangle |
| itemList | QwtSymbol, 579 |
| QwtPlotDict, 317 | label |
| Q | 14001 |

| QwtAbstractScaleDraw, 47 | LegendBackground |
|------------------------------|------------------------------------|
| QwtCompassScaleDraw, 86 | QwtPlotLegendItem, 375 |
| QwtDateScaleDraw, 107 | LegendBarTitles |
| QwtPlotMarker, 386 | QwtPlotBarChart, 291 |
| labelAlignment | LegendChartTitle |
| QwtPlotMarker, 387 | QwtPlotBarChart, 291 |
| QwtScaleDraw, 524 | LegendColor |
| labelMap | QwtPlotShapeItem, 439 |
| QwtCompassScaleDraw, 86 | LegendInterest |
| labelOrientation | QwtPlotItem, 352 |
| QwtPlotMarker, 387 | LegendNoAttribute |
| labelPosition | QwtPlotCurve, 305 |
| QwtScaleDraw, 524 | LegendShape |
| labelRect | QwtPlotShapeItem, 439 |
| QwtArrowButton, 71 | LegendShowBrush |
| QwtScaleDraw, 525 | QwtPlotCurve, 305 |
| labelRotation | LegendShowLine |
| QwtScaleDraw, 525 | QwtPlotCurve, 305 |
| labelSize | LegendShowSymbol |
| QwtScaleDraw, 525 | QwtPlotCurve, 305 |
| labelTransformation | LegendAttribute |
| QwtScaleDraw, 525 Labels | QwtPlotCurve, 305 |
| | legendChanged |
| QwtAbstractScaleDraw, 44 | QwtPlotItem, 355 |
| LayoutAttribute QwtText, 594 | legendData QwtPlotBarChart, 293 |
| LayoutFlag | QwtPlotItem, 355 |
| QwtPlotRenderer, 413 | QwtPlotMultiBarChart, 394 |
| QwtScaleWidget, 542 | legendDataChanged |
| layoutFlags | QwtPlot, 277 |
| QwtPlotRenderer, 414 | legendGeometries |
| layoutGrid | QwtPlotLegendItem, 377 |
| QwtDynGridLayout, 131 | legendicon |
| layoutHint | QwtPlotBarChart, 293 |
| QwtPlotAbstractBarChart, 287 | QwtPlotCurve, 310 |
| layoutltems | QwtPlotHistogram, 338 |
| QwtDynGridLayout, 131 | QwtPlotIntervalCurve, 345 |
| layoutLegend | QwtPlotItem, 355 |
| QwtPlotLayout, 367 | QwtPlotMarker, 387 |
| LayoutPolicy | QwtPlotMultiBarChart, 394 |
| QwtPlotAbstractBarChart, 286 | QwtPlotShapeItem, 440 |
| layoutPolicy | QwtPlotTradingCurve, 468 |
| QwtPlotAbstractBarChart, 287 | legendlconSize |
| layoutScale | QwtPlotItem, 356 |
| QwtScaleWidget, 545 | LegendMode |
| LeadingScale | QwtPlotBarChart, 291 |
| QwtSlider, 564 | QwtPlotShapeItem, 439 |
| QwtThermo, 610 | legendMode |
| LeftLegend | QwtPlotBarChart, 294 |
| QwtPlot, 268 | QwtPlotShapeltem, 440 |
| LeftScale | LegendPosition |
| QwtScaleDraw, 522 | QwtPlot, 267 |
| LeftToRight | legendPosition |
| QwtColumnRect, 76 | QwtPlotLayout, 367 |
| Legend | legendRatio |
| QwtPlotItem, 351 | QwtPlotLayout, 367 |
| legend | legendRect |
| QwtPlot, 277 | QwtPlotLayout, 368 |
| | • • |

| logopd\\/idget | OutToxt FOE |
|---|--|
| legendWidget | QwtText, 595 |
| QwtLegend, 180 | maxColumns |
| legendWidgets | QwtDynGridLayout, 132 |
| QwtLegend, 180 | QwtLegend, 180 |
| length | QwtPlotLegendItem, 377 |
| QwtScaleDraw, 525 | maxDate |
| limited | QwtDate, 102 |
| QwtInterval, 155 | maxItemWidth |
| linePen | QwtDynGridLayout, 132 |
| QwtPlotMarker, 387 | maxLabelHeight |
| LineStyle | QwtScaleDraw, 526 |
| QwtPlotMarker, 385 | maxLabelWidth |
| lineStyle | QwtScaleDraw, 526 |
| QwtPlotMarker, 387 | maxScaleArc |
| lineWidth | QwtDial, 120 |
| QwtColumnSymbol, 78 | maxStackDepth |
| QwtDial, 120 | QwtPlotZoomer, 481 |
| QwtPlotGLCanvas, 325 | maxSymbolWidth |
| Lines | - |
| QwtPlotCurve, 305 | QwtPlotTradingCurve, 469 |
| QwtPlotHistogram, 334 | maxTickLength |
| loadData | QwtAbstractScaleDraw, 47 |
| | maxValue |
| QwtPlotSvgItem, 458 | QwtInterval, 156 |
| loadFile | maxWeeks |
| QwtPlotSvgItem, 459 | QwtDateScaleEngine, 112 |
| lowerBound | maximum |
| QwtAbstractScale, 37 | QwtAbstractScale, 37 |
| QwtScaleDiv, 518 | QwtCounter, 92 |
| lowerMargin | QwtWheel, 628 |
| QwtScaleEngine, 533 | MediumTick |
| M : T: I | QwtScaleDiv, 515 |
| MajorTick | metric |
| QwtScaleDiv, 515 | QwtNullPaintDevice, 216 |
| | |
| majorPen | |
| majorPen QwtPlotGrid, 329 | midLineWidth |
| majorPen QwtPlotGrid, 329 margin | midLineWidth QwtPlotGLCanvas, 325 |
| majorPen QwtPlotGrid, 329 | midLineWidth QwtPlotGLCanvas, 325 mightRender |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins QwtPlotItem, 351 | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins QwtPlotItem, 351 markerSize | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond QwtDate, 101 |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins QwtPlotItem, 351 markerSize QwtKnob, 171 | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond QwtDate, 101 minDate |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins QwtPlotItem, 351 markerSize QwtKnob, 171 MarkerStyle QwtKnob, 169 | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond QwtDate, 101 minDate QwtDate, 102 |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins QwtPlotItem, 351 markerSize QwtKnob, 171 MarkerStyle QwtKnob, 169 markerStyle | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond QwtDate, 101 minDate QwtDate, 102 minLabelDist |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins QwtPlotItem, 351 markerSize QwtKnob, 171 MarkerStyle QwtKnob, 169 markerStyle QwtKnob, 171 | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond QwtDate, 101 minDate QwtDate, 102 minLabelDist QwtScaleDraw, 526 minLength |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins QwtPlotItem, 351 markerSize QwtKnob, 171 MarkerStyle QwtKnob, 169 markerStyle QwtKnob, 171 MaskHint | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond QwtDate, 101 minDate QwtDate, 102 minLabelDist QwtScaleDraw, 526 minLength QwtScaleDraw, 526 |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins QwtPlotItem, 351 markerSize QwtKnob, 171 MarkerStyle QwtKnob, 169 markerStyle QwtKnob, 171 MaskHint QwtWidgetOverlay, 639 | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond QwtDate, 101 minDate QwtDate, 102 minLabelDist QwtScaleDraw, 526 minLength QwtScaleDraw, 526 minScaleArc |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins QwtPlotItem, 351 markerSize QwtKnob, 171 MarkerStyle QwtKnob, 169 markerStyle QwtKnob, 171 MaskHint QwtWidgetOverlay, 639 maskHint | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond QwtDate, 101 minDate QwtDate, 102 minLabelDist QwtScaleDraw, 526 minLength QwtScaleDraw, 526 minScaleArc QwtDial, 120 |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins QwtPlotItem, 351 markerSize QwtKnob, 171 MarkerStyle QwtKnob, 169 markerStyle QwtKnob, 171 MaskHint QwtWidgetOverlay, 639 maskHint QwtWidgetOverlay, 640 | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond QwtDate, 101 minDate QwtDate, 102 minLabelDist QwtScaleDraw, 526 minLength QwtScaleDraw, 526 minScaleArc QwtDial, 120 minSymbolWidth |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins QwtPlotItem, 351 markerSize QwtKnob, 171 MarkerStyle QwtKnob, 169 markerStyle QwtKnob, 171 MaskHint QwtWidgetOverlay, 639 maskHint QwtWidgetOverlay, 640 MaskMode | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond QwtDate, 101 minDate QwtDate, 102 minLabelDist QwtScaleDraw, 526 minLength QwtScaleDraw, 526 minScaleArc QwtDial, 120 minSymbolWidth QwtPlotTradingCurve, 469 |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins QwtPlotItem, 351 markerSize QwtKnob, 171 MarkerStyle QwtKnob, 169 markerStyle QwtKnob, 171 MaskHint QwtWidgetOverlay, 639 maskHint QwtWidgetOverlay, 640 MaskMode QwtWidgetOverlay, 639 | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond QwtDate, 101 minDate QwtDate, 102 minLabelDist QwtScaleDraw, 526 minLength QwtScaleDraw, 526 minScaleArc QwtDial, 120 minSymbolWidth QwtPlotTradingCurve, 469 minValue |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins QwtPlotItem, 351 markerSize QwtKnob, 171 MarkerStyle QwtKnob, 169 markerStyle QwtKnob, 171 MaskHint QwtWidgetOverlay, 639 maskHint QwtWidgetOverlay, 640 MaskMode QwtWidgetOverlay, 639 maskMode | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond QwtDate, 101 minDate QwtDate, 102 minLabelDist QwtScaleDraw, 526 minLength QwtScaleDraw, 526 minScaleArc QwtDial, 120 minSymbolWidth QwtPlotTradingCurve, 469 minValue QwtInterval, 156 |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins QwtPlotItem, 351 markerSize QwtKnob, 171 MarkerStyle QwtKnob, 169 markerStyle QwtKnob, 171 MaskHint QwtWidgetOverlay, 639 maskHint QwtWidgetOverlay, 640 MaskMode QwtWidgetOverlay, 639 maskMode QwtWidgetOverlay, 639 maskMode QwtWidgetOverlay, 639 maskMode QwtWidgetOverlay, 640 | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond QwtDate, 101 minDate QwtDate, 102 minLabelDist QwtScaleDraw, 526 minLength QwtScaleDraw, 526 minScaleArc QwtDial, 120 minSymbolWidth QwtPlotTradingCurve, 469 minValue QwtInterval, 156 minZoomSize |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins QwtPlotItem, 351 markerSize QwtKnob, 171 MarkerStyle QwtKnob, 169 markerStyle QwtKnob, 171 MaskHint QwtWidgetOverlay, 639 maskHint QwtWidgetOverlay, 640 MaskMode QwtWidgetOverlay, 639 maskMode QwtWidgetOverlay, 639 maskMode QwtWidgetOverlay, 640 mass | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond QwtDate, 101 minDate QwtDate, 102 minLabelDist QwtScaleDraw, 526 minLength QwtScaleDraw, 526 minScaleArc QwtDial, 120 minSymbolWidth QwtPlotTradingCurve, 469 minValue QwtInterval, 156 minZoomSize QwtPlotZoomer, 481 |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins QwtPlotItem, 351 markerSize QwtKnob, 171 MarkerStyle QwtKnob, 169 markerStyle QwtKnob, 171 MaskHint QwtWidgetOverlay, 639 maskHint QwtWidgetOverlay, 640 MaskMode QwtWidgetOverlay, 639 maskMode QwtWidgetOverlay, 640 mass QwtWheel, 628 | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond QwtDate, 101 minDate QwtDate, 102 minLabelDist QwtScaleDraw, 526 minLength QwtScaleDraw, 526 minScaleArc QwtDial, 120 minSymbolWidth QwtPlotTradingCurve, 469 minValue QwtInterval, 156 minZoomSize QwtPlotZoomer, 481 MinimizeMemory |
| majorPen QwtPlotGrid, 329 margin QwtPlotAbstractBarChart, 287 QwtPlotLegendItem, 377 QwtPlotTextLabel, 462 QwtScaleWidget, 545 Margins QwtPlotItem, 351 markerSize QwtKnob, 171 MarkerStyle QwtKnob, 169 markerStyle QwtKnob, 171 MaskHint QwtWidgetOverlay, 639 maskHint QwtWidgetOverlay, 640 MaskMode QwtWidgetOverlay, 639 maskMode QwtWidgetOverlay, 639 maskMode QwtWidgetOverlay, 640 mass | midLineWidth QwtPlotGLCanvas, 325 mightRender QwtMathMLTextEngine, 210 QwtPlainTextEngine, 263 QwtRichTextEngine, 507 QwtTextEngine, 602 Millisecond QwtDate, 101 minDate QwtDate, 102 minLabelDist QwtScaleDraw, 526 minLength QwtScaleDraw, 526 minScaleArc QwtDial, 120 minSymbolWidth QwtPlotTradingCurve, 469 minValue QwtInterval, 156 minZoomSize QwtPlotZoomer, 481 |

| minimum | QwtWheel, 628 |
|--|---|
| QwtAbstractScale, 38 | mousePattern |
| QwtCounter, 92 | QwtEventPattern, 139 |
| QwtWheel, 628 | MousePatternCode |
| MinimumLayout | QwtEventPattern, 136 |
| QwtText, 594 | mousePressEvent |
| minimumExtent | QwtAbstractSlider, 56 |
| QwtAbstractScaleDraw, 47 | QwtSlider, 566 |
| minimumSize | QwtWheel, 629 |
| QwtPlotLegendItem, 378 | mouseReleaseEvent |
| minimumSizeHint | QwtAbstractSlider, 56 |
| QwtDial, 120 | QwtSlider, 566 |
| QwtKnob, 171 | QwtWheel, 629 |
| QwtPlotLayout, 368 | move |
| QwtScaleWidget, 545 | QwtPicker, 242 |
| QwtSlider, 566 | QwtPlotPicker, 404 |
| QwtThermo, 613 | QwtScaleDraw, 526, 528 |
| QwtWheel, 628 | moveBy |
| MinorTick | QwtPlotZoomer, 481 |
| QwtScaleDiv, 515 | moveCanvas |
| minorPen | QwtPlotPanner, 400 |
| | moveCenter |
| QwtPlotGrid, 329 | QwtRoundScaleDraw, 510 |
| Minute | moveTo |
| QwtDate, 101 | QwtPlotZoomer, 482 |
| MinuteHand | moved |
| QwtAnalogClock, 64 | QwtPanner, 231 |
| Mode | QwtPicker, 242 |
| QwtDial, 117 | QwtPlotPicker, 404 |
| QwtLegendData, 183 | QWIFIDIFICKEI, 404 |
| _ | |
| QwtLinearColorMap, 190 | NHands |
| _ | NHands OwtAnalogClock, 64 |
| QwtLinearColorMap, 190 | QwtAnalogClock, 64 |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 | QwtAnalogClock, 64 NTickTypes |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 MouseSelect1 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 NoCache |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 MouseSelect1 QwtEventPattern, 136 MouseSelect2 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 NoCache QwtPlotRasterItem, 408 |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 MouseSelect1 QwtEventPattern, 136 MouseSelect2 QwtEventPattern, 136 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 NoCache QwtPlotRasterItem, 408 QwtSymbol, 578 |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 MouseSelect1 QwtEventPattern, 136 MouseSelect2 QwtEventPattern, 136 MouseSelect3 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 NoCache QwtPlotRasterItem, 408 QwtSymbol, 578 NoCurve |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 MouseSelect1 QwtEventPattern, 136 MouseSelect2 QwtEventPattern, 136 MouseSelect3 QwtEventPattern, 136 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 NoCache QwtPlotRasterItem, 408 QwtSymbol, 578 NoCurve QwtPlotCurve, 305 |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 MouseSelect1 QwtEventPattern, 136 MouseSelect2 QwtEventPattern, 136 MouseSelect3 QwtEventPattern, 136 MouseSelect4 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 NoCache QwtPlotRasterItem, 408 QwtSymbol, 578 NoCurve QwtPlotCurve, 305 QwtPlotIntervalCurve, 343 |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 MouseSelect1 QwtEventPattern, 136 MouseSelect2 QwtEventPattern, 136 MouseSelect3 QwtEventPattern, 136 MouseSelect4 QwtEventPattern, 136 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 NoCache QwtPlotRasterItem, 408 QwtSymbol, 578 NoCurve QwtPlotCurve, 305 QwtPlotIntervalCurve, 343 NoFocusIndicator |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 MouseSelect1 QwtEventPattern, 136 MouseSelect2 QwtEventPattern, 136 MouseSelect3 QwtEventPattern, 136 MouseSelect4 QwtEventPattern, 136 MouseSelect4 QwtEventPattern, 136 MouseSelect5 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 NoCache QwtPlotRasterItem, 408 QwtSymbol, 578 NoCurve QwtPlotCurve, 305 QwtPlotIntervalCurve, 343 NoFocusIndicator QwtPlotCanvas, 298 |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 MouseSelect1 QwtEventPattern, 136 MouseSelect2 QwtEventPattern, 136 MouseSelect3 QwtEventPattern, 136 MouseSelect4 QwtEventPattern, 136 MouseSelect5 QwtEventPattern, 136 MouseSelect5 QwtEventPattern, 136 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 NoCache QwtPlotRasterItem, 408 QwtSymbol, 578 NoCurve QwtPlotCurve, 305 QwtPlotIntervalCurve, 343 NoFocusIndicator QwtPlotCanvas, 298 NoFrame |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 MouseSelect1 QwtEventPattern, 136 MouseSelect2 QwtEventPattern, 136 MouseSelect3 QwtEventPattern, 136 MouseSelect4 QwtEventPattern, 136 MouseSelect5 QwtEventPattern, 136 MouseSelect5 QwtEventPattern, 136 MouseSelect5 QwtEventPattern, 136 MouseSelect6 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 NoCache QwtPlotRasterItem, 408 QwtSymbol, 578 NoCurve QwtPlotCurve, 305 QwtPlotIntervalCurve, 343 NoFocusIndicator QwtPlotCanvas, 298 NoFrame QwtColumnSymbol, 77 |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 MouseSelect1 QwtEventPattern, 136 MouseSelect2 QwtEventPattern, 136 MouseSelect3 QwtEventPattern, 136 MouseSelect4 QwtEventPattern, 136 MouseSelect5 QwtEventPattern, 136 MouseSelect5 QwtEventPattern, 136 MouseSelect6 QwtEventPattern, 136 MouseSelect6 QwtEventPattern, 137 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 NoCache QwtPlotRasterItem, 408 QwtSymbol, 578 NoCurve QwtPlotCurve, 305 QwtPlotIntervalCurve, 343 NoFocusIndicator QwtPlotCanvas, 298 NoFrame QwtColumnSymbol, 77 NoLine |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 MouseSelect1 QwtEventPattern, 136 MouseSelect2 QwtEventPattern, 136 MouseSelect3 QwtEventPattern, 136 MouseSelect4 QwtEventPattern, 136 MouseSelect5 QwtEventPattern, 136 MouseSelect5 QwtEventPattern, 136 MouseSelect6 QwtEventPattern, 137 mouseFactor | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 NoCache QwtPlotRasterItem, 408 QwtSymbol, 578 NoCurve QwtPlotCurve, 305 QwtPlotIntervalCurve, 343 NoFocusIndicator QwtPlotCanvas, 298 NoFrame QwtColumnSymbol, 77 NoLine QwtPlotMarker, 385 |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 MouseSelect1 QwtEventPattern, 136 MouseSelect2 QwtEventPattern, 136 MouseSelect3 QwtEventPattern, 136 MouseSelect4 QwtEventPattern, 136 MouseSelect5 QwtEventPattern, 136 MouseSelect6 QwtEventPattern, 136 MouseSelect6 QwtEventPattern, 137 mouseFactor QwtMagnifier, 203 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 NoCache QwtPlotRasterItem, 408 QwtSymbol, 578 NoCurve QwtPlotCurve, 305 QwtPlotIntervalCurve, 343 NoFocusIndicator QwtPlotCanvas, 298 NoFrame QwtColumnSymbol, 77 NoLine QwtPlotMarker, 385 NoMarker |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 MouseSelect1 QwtEventPattern, 136 MouseSelect2 QwtEventPattern, 136 MouseSelect3 QwtEventPattern, 136 MouseSelect4 QwtEventPattern, 136 MouseSelect5 QwtEventPattern, 136 MouseSelect5 QwtEventPattern, 136 MouseSelect6 QwtEventPattern, 137 mouseFactor QwtMagnifier, 203 mouseMatch | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 NoCache QwtPlotRasterItem, 408 QwtSymbol, 578 NoCurve QwtPlotCurve, 305 QwtPlotIntervalCurve, 343 NoFocusIndicator QwtPlotCanvas, 298 NoFrame QwtColumnSymbol, 77 NoLine QwtPlotMarker, 385 NoMarker QwtKnob, 169 |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 MouseSelect1 QwtEventPattern, 136 MouseSelect2 QwtEventPattern, 136 MouseSelect3 QwtEventPattern, 136 MouseSelect4 QwtEventPattern, 136 MouseSelect5 QwtEventPattern, 136 MouseSelect6 QwtEventPattern, 136 MouseSelect6 QwtEventPattern, 137 mouseFactor QwtMagnifier, 203 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 NoCache QwtPlotRasterItem, 408 QwtSymbol, 578 NoCurve QwtPlotCurve, 305 QwtPlotIntervalCurve, 343 NoFocusIndicator QwtPlotCanvas, 298 NoFrame QwtColumnSymbol, 77 NoLine QwtPlotMarker, 385 NoMarker QwtKnob, 169 NoMask |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 MouseSelect1 QwtEventPattern, 136 MouseSelect2 QwtEventPattern, 136 MouseSelect3 QwtEventPattern, 136 MouseSelect4 QwtEventPattern, 136 MouseSelect5 QwtEventPattern, 136 MouseSelect5 QwtEventPattern, 136 MouseSelect6 QwtEventPattern, 137 mouseFactor QwtMagnifier, 203 mouseMatch | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 NoCache QwtPlotRasterItem, 408 QwtSymbol, 578 NoCurve QwtPlotCurve, 305 QwtPlotIntervalCurve, 343 NoFocusIndicator QwtPlotCanvas, 298 NoFrame QwtColumnSymbol, 77 NoLine QwtPlotMarker, 385 NoMarker QwtKnob, 169 |
| QwtLinearColorMap, 190 QwtNullPaintDevice, 216 mode QwtDial, 120 QwtLegendData, 184 QwtLinearColorMap, 192 QwtNullPaintDevice, 216 Month QwtDate, 101 MousePatternCount QwtEventPattern, 137 MouseSelect1 QwtEventPattern, 136 MouseSelect2 QwtEventPattern, 136 MouseSelect3 QwtEventPattern, 136 MouseSelect4 QwtEventPattern, 136 MouseSelect5 QwtEventPattern, 136 MouseSelect6 QwtEventPattern, 136 MouseSelect6 QwtEventPattern, 137 mouseFactor QwtMagnifier, 203 mouseMatch QwtEventPattern, 138 | QwtAnalogClock, 64 NTickTypes QwtScaleDiv, 515 Natural QwtSpline, 572 NearestNeighbour QwtMatrixRasterData, 212 needle QwtDial, 120, 121 NoAttribute QwtScaleEngine, 531 NoCache QwtPlotRasterItem, 408 QwtSymbol, 578 NoCurve QwtPlotCurve, 305 QwtPlotIntervalCurve, 343 NoFocusIndicator QwtPlotCanvas, 298 NoFrame QwtColumnSymbol, 77 NoLine QwtPlotMarker, 385 NoMarker QwtKnob, 169 NoMask |

| 0.1514 | 0.51.5 |
|---------------------------|---------------------------|
| QwtPicker, 238 | QwtPlotZoneItem, 476 |
| NoScale | QwtScaleDraw, 528 |
| QwtSlider, 564 | QwtSlider, 566 |
| QwtThermo, 610 | QwtThermo, 613 |
| NoSelection | QwtWheel, 629 |
| QwtPickerMachine, 257 | origin |
| NoStyle | QwtDial, 121 |
| QwtColumnSymbol, 77 | QwtThermo, 613 |
| NoSymbol | OriginCustom |
| QwtIntervalSymbol, 164 | QwtThermo, 609 |
| QwtPlotTradingCurve, 466 | OriginMaximum |
| QwtSymbol, 579 | QwtThermo, 609 |
| NoTick | OriginMinimum |
| QwtScaleDiv, 515 | QwtThermo, 609 |
| NormalMode | OriginMode |
| QwtNullPaintDevice, 216 | QwtThermo, 609 |
| normalized | originMode |
| QwtInterval, 156 | QwtThermo, 614 |
| QwtPointPolar, 497 | OtherFormat |
| Notch | QwtText, 595 |
| QwtKnob, 169 | Outline |
| Nub | QwtPlotHistogram, 334 |
| QwtKnob, 169 | ann ion iongiani, oo |
| numButtons | p1 |
| QwtCounter, 93 | QwtScaleMap, 538 |
| numColumns | p2 |
| QwtDynGridLayout, 132 | QwtScaleMap, 538 |
| QwtMatrixRasterData, 212 | pDist |
| numRows | QwtScaleMap, 538 |
| | pageStepCount |
| QwtDynGridLayout, 132 | QwtWheel, 629 |
| QwtMatrixRasterData, 212 | pageSteps |
| numThornLevels | QwtAbstractSlider, 56 |
| QwtSimpleCompassRose, 560 | PaintBackground |
| numThorns | QwtText, 594 |
| QwtSimpleCompassRose, 561 | PaintCache |
| numTurns | QwtPlotRasterItem, 408 |
| QwtKnob, 171 | PaintInDeviceResolution |
| Opaqua | QwtPlotRasterItem, 408 |
| Opaque OutPlotConvos 208 | PaintUsingTextColor |
| QwtPlotCanvas, 298 | QwtText, 594 |
| operator= | • |
| QwtGraphic, 146 | PaintUsingTextFont |
| QwtPainterCommand, 227 | QwtText, 594 |
| QwtSpline, 572 | PaintAttribute |
| operator== | QwtPlotCanvas, 298 |
| QwtInterval, 158 | QwtPlotCurve, 305 |
| QwtPoint3D, 487 | QwtPlotIntervalCurve, 343 |
| QwtPointPolar, 498 | QwtPlotRasterItem, 408 |
| QwtScaleDiv, 519 | QwtPlotShapeItem, 439 |
| operator& | QwtPlotSpectroCurve, 445 |
| QwtInterval, 156 | QwtPlotTradingCurve, 466 |
| operator&= | QwtText, 594 |
| QwtInterval, 156 | paintEvent |
| Option | QwtArrowButton, 71 |
| QwtPlotLayout, 364 | QwtDial, 121 |
| orientation | QwtKnob, 171 |
| QwtColumnRect, 76 | QwtPanner, 232 |
| QwtPlotRescaler, 425 | QwtPlotCanvas, 300 |
| QwtPlotSeriesItem, 437 | QwtPlotGLCanvas, 325 |
| , | , |

| QwtSlider, 567 | QwtSymbol, 582 |
|------------------------------------|-------------------------|
| QwtTextLabel, 605 | pixmapData |
| QwtThermo, 614 | QwtPainterCommand, 228 |
| QwtWheel, 629 | Plain |
| QwtWidgetOverlay, 641 | QwtColumnSymbol, 77 |
| paintRect | QwtDial, 117 |
| QwtPlotItem, 356 | QwtPlotGLCanvas, 323 |
| palette | PlainText |
| QwtColumnSymbol, 78 | QwtText, 595 |
| QwtCompassRose, 85 | plainText |
| QwtDialNeedle, 126 | QwtTextLabel, 605 |
| QwtPlotScaleItem, 432 | plot |
| panned | QwtPlotPicker, 405 |
| QwtPanner, 232 | QwtPlotRescaler, 425 |
| ParametricSpline | plotItems |
| QwtSplineCurveFitter, 575 | QwtPlotLegendItem, 378 |
| parentWidget | plotLayout |
| QwtMagnifier, 203 | QwtPlot, 278 |
| Path | PointSelection |
| QwtPainterCommand, 226 | QwtPickerMachine, 257 |
| QwtSymbol, 579 | points |
| | QwtSpline, 573 |
| path OutBainterCommand 228 | PolygonPathMode |
| QwtPainterCommand, 228 | QwtNullPaintDevice, 216 |
| QwtSymbol, 582 | PolygonRubberBand |
| PathMode | QwtPicker, 238 |
| QwtNullPaintDevice, 216 | PolygonSelection |
| pen College December 100 de la 105 | QwtPickerMachine, 257 |
| QwtIntervalSymbol, 165 | polylineSplitting |
| QwtPlotCurve, 311 | QwtPainter, 224 |
| QwtPlotHistogram, 338 | pos |
| QwtPlotIntervalCurve, 345 | QwtScaleDraw, 528 |
| QwtPlotShapeItem, 441 | position |
| QwtPlotZoneItem, 476 | QwtPlotScaleItem, 432 |
| QwtSymbol, 582 | GWI lotodalottom, 102 |
| penWidth | QwtAbstractScaleDraw |
| QwtAbstractScaleDraw, 47 | Backbone, 44 |
| QwtPlotSpectroCurve, 446 | Labels, 44 |
| Periodic | Ticks, 44 |
| QwtSpline, 572 | QwtAnalogClock |
| pickArea | HourHand, 64 |
| QwtPicker, 242 | MinuteHand, 64 |
| pickedPoints | NHands, 64 |
| QwtPicker, 242 | SecondHand, 64 |
| pinPoint | QwtColorMap |
| QwtSymbol, 582 | Indexed, 74 |
| pipeRect | RGB, 74 |
| QwtThermo, 614 | QwtColumnRect |
| pipeWidth | BottomToTop, 76 |
| QwtThermo, 614 | LeftToRight, 76 |
| pixelHint | RightToLeft, 76 |
| QwtMatrixRasterData, 212 | TopToBottom, 76 |
| QwtPlotRasterItem, 410 | QwtColumnSymbol |
| QwtPlotSpectrogram, 453 | Box, 77 |
| QwtRasterData, 504 | NoFrame, 77 |
| Pixmap | NoStyle, 77 |
| QwtPainterCommand, 226 | Plain, 77 |
| QwtSymbol, 579 | Raised, 77 |
| pixmap | UserStyle, 77 |
| ριλιτιαρ | OserStyle, 77 |

| QwtCompassMagnetNeedle | QwtIntervalSymbol |
|-------------------------|----------------------------|
| ThinStyle, 83 | Bar, 164 |
| TriangleStyle, 83 | Box, 164 |
| QwtCompassWindArrow | NoSymbol, 164 |
| Style1, 88 | UserSymbol, 164 |
| Style2, 88 | QwtKnob |
| QwtCounter | Dot, 169 |
| | |
| Button1, 90 | Flat, 169 |
| Button2, 90 | NoMarker, 169 |
| Button3, 91 | Notch, 169 |
| ButtonCnt, 91 | Nub, 169 |
| QwtDate | Raised, 169 |
| Day, 101 | Styled, 169 |
| FirstDay, 101 | Sunken, 169 |
| FirstThursday, 101 | Tick, 169 |
| Hour, 101 | Triangle, 169 |
| JulianDayForEpoch, 100 | QwtLegendData |
| Millisecond, 101 | Checkable, 183 |
| Minute, 101 | Clickable, 183 |
| Month, 101 | ReadOnly, 183 |
| Second, 101 | QwtLinearColorMap |
| Week, 101 | FixedColors, 191 |
| Year, 101 | ScaledColors, 191 |
| QwtDial | QwtMatrixRasterData |
| Plain, 117 | BilinearInterpolation, 212 |
| Raised, 117 | NearestNeighbour, 212 |
| | QwtNullPaintDevice |
| RotateNeedle, 117 | |
| RotateScale, 117 | NormalMode, 216 |
| Sunken, 117 | PathMode, 216 |
| QwtDialSimpleNeedle | PolygonPathMode, 216 |
| Arrow, 127 | QwtPainterCommand |
| Ray, 127 | Image, 226 |
| QwtEventPattern | Invalid, 226 |
| KeyAbort, 136 | Path, 226 |
| KeyDown, 136 | Pixmap, 226 |
| KeyHome, 136 | State, 226 |
| KeyLeft, 136 | QwtPicker |
| KeyPatternCount, 136 | ActiveOnly, 237 |
| KeyRedo, 136 | AlwaysOff, 237 |
| KeyRight, 136 | AlwaysOn, 237 |
| KeySelect1, 136 | CrossRubberBand, 238 |
| KeySelect2, 136 | EllipseRubberBand, 238 |
| KeyUndo, 136 | HLineRubberBand, 238 |
| KeyUp, 136 | KeepSize, 238 |
| MousePatternCount, 137 | NoRubberBand, 238 |
| MouseSelect1, 136 | PolygonRubberBand, 238 |
| MouseSelect2, 136 | RectRubberBand, 238 |
| MouseSelect3, 136 | Stretch, 238 |
| | |
| MouseSelect4, 136 | UserRubberBand, 238 |
| MouseSelect5, 136 | VLineRubberBand, 238 |
| MouseSelect6, 137 | QwtPickerMachine |
| QwtGraphic | NoSelection, 257 |
| RenderPensUnscaled, 142 | PointSelection, 257 |
| QwtInterval | PolygonSelection, 257 |
| ExcludeBorders, 153 | RectSelection, 257 |
| ExcludeMaximum, 153 | QwtPlot |
| ExcludeMinimum, 153 | axisCnt, 267 |
| IncludeBorders, 153 | BottomLegend, 268 |
| | |

| LeftLegend, 268 | UserCurve, 343 |
|---------------------------|------------------------------|
| RightLegend, 268 | QwtPlotItem |
| TopLegend, 268 | AutoScale, 351 |
| xBottom, 267 | Legend, 351 |
| xTop, 267 | LegendInterest, 352 |
| yLeft, 267 | Margins, 351 |
| yRight, 267 | RenderAntialiased, 352 |
| QwtPlotAbstractBarChart | Rtti_PlotBarChart, 352 |
| AutoAdjustSamples, 286 | Rtti_PlotCurve, 352 |
| FixedSampleSize, 286 | Rtti_PlotGrid, 352 |
| ScaleSampleToCanvas, 286 | Rtti_PlotHistogram, 352 |
| ScaleSamplesToAxes, 286 | Rtti_PlotIntervalCurve, 352 |
| QwtPlotBarChart | Rtti_PlotItem, 352 |
| LegendBarTitles, 291 | Rtti_PlotLegend, 352 |
| LegendChartTitle, 291 | Rtti PlotMarker, 352 |
| QwtPlotCanvas | Rtti PlotMultiBarChart, 353 |
| BackingStore, 298 | Rtti_PlotSVG, 352 |
| CanvasFocusIndicator, 298 | Rtti PlotScale, 352 |
| HackStyledBackground, 298 | Rtti_PlotShape, 353 |
| ImmediatePaint, 298 | Rtti_PlotSpectroCurve, 352 |
| ItemFocusIndicator, 298 | Rtti_PlotSpectrogram, 352 |
| NoFocusIndicator, 298 | Rtti PlotTextLabel, 353 |
| Opaque, 298 | Rtti PlotTradingCurve, 352 |
| QwtPlotCurve | Rtti PlotUserItem, 353 |
| ClipPolygons, 305 | Rtti_PlotZone, 353 |
| Dots, 305 | ScaleInterest, 352 |
| FilterPoints, 305 | QwtPlotLayout |
| Fitted, 304 | AlignScales, 365 |
| ImageBuffer, 305 | IgnoreFooter, 365 |
| Inverted, 304 | IgnoreFrames, 365 |
| LegendNoAttribute, 305 | IgnoreLegend, 365 |
| LegendShowBrush, 305 | IgnoreScrollbars, 365 |
| LegendShowLine, 305 | IgnoreTitle, 365 |
| LegendShowSymbol, 305 | QwtPlotLegendItem |
| Lines, 305 | ItemBackground, 375 |
| MinimizeMemory, 305 | LegendBackground, 375 |
| NoCurve, 305 | QwtPlotMarker |
| Steps, 305 | Cross, 385 |
| Sticks, 305 | HLine, 385 |
| UserCurve, 305 | NoLine, 385 |
| QwtPlotDirectPainter | VLine, 385 |
| AtomicPainter, 320 | QwtPlotMultiBarChart |
| CopyBackingStore, 320 | Grouped, 392 |
| FullRepaint, 320 | Stacked, 392 |
| QwtPlotGLCanvas | QwtPlotRasterItem |
| Plain, 323 | NoCache, 408 |
| Raised, 323 | PaintCache, 408 |
| Sunken, 323 | PaintInDeviceResolution, 408 |
| QwtPlotHistogram | QwtPlotRenderer |
| Columns, 334 | DefaultLayout, 414 |
| Lines, 334 | DiscardBackground, 413 |
| Outline, 334 | DiscardCanvasBackground, 413 |
| UserStyle, 334 | DiscardCanvasFrame, 413 |
| QwtPlotIntervalCurve | DiscardFooter, 413 |
| ClipPolygons, 343 | DiscardLegend, 413 |
| ClipSymbol, 343 | DiscardNone, 413 |
| NoCurve, 343 | DiscardTitle, 413 |
| Tube, 343 | FrameWithScales, 414 |
| 1000, 070 | Trainie With Ocales, 414 |

| QwtPlotRescaler | ParametricSpline, 575 |
|-------------------------------|--|
| ExpandBoth, 421 | Spline, 575 |
| ExpandDown, 421 | QwtSymbol |
| ExpandUp, 421 | AutoCache, 578 |
| Expanding, 421 | Cache, 578 |
| Fitting, 421 | Cross, 579 |
| Fixed, 421 | DTriangle, 579 |
| | |
| QwtPlotShapeItem | Diamond, 579 |
| ClipPolygons, 439 | Ellipse, 579 |
| LegendColor, 439 | Graphic, 579 |
| LegendShape, 439 | HLine, 579 |
| QwtPlotSpectroCurve | Hexagon, 579 |
| ClipPoints, 445 | LTriangle, 579 |
| QwtPlotSpectrogram | NoCache, 578 |
| ContourMode, 450 | NoSymbol, 579 |
| ImageMode, 450 | Path, 579 |
| QwtPlotTradingCurve | Pixmap, 579 |
| Bar, 466 | RTriangle, 579 |
| CandleStick, 466 | Rect, 579 |
| ClipSymbols, 466 | Star1, 579 |
| • • | |
| Decreasing, 466 | Star2, 579 |
| Increasing, 466 | SvgDocument, 579 |
| NoSymbol, 466 | Triangle, 579 |
| UserSymbol, 466 | UTriangle, 579 |
| QwtPointMapper | UserStyle, 579 |
| RoundPoints, 492 | VLine, 579 |
| WeedOutPoints, 492 | XCross, 579 |
| QwtRasterData | QwtText |
| IgnoreAllVerticesOnLevel, 503 | AutoText, 595 |
| IgnoreOutOfRange, 503 | MathMLText, 595 |
| QwtScaleDiv | MinimumLayout, 594 |
| MajorTick, 515 | OtherFormat, 595 |
| MediumTick, 515 | PaintBackground, 594 |
| MinorTick, 515 | PaintUsingTextColor, 594 |
| NTickTypes, 515 | PaintUsingTextFont, 594 |
| | • |
| NoTick, 515 | PlainText, 595 |
| QwtScaleDraw | RichText, 595 |
| BottomScale, 522 | TeXText, 595 |
| LeftScale, 522 | QwtThermo |
| RightScale, 522 | LeadingScale, 610 |
| TopScale, 522 | NoScale, 610 |
| QwtScaleEngine | OriginCustom, 609 |
| Floating, 531 | OriginMaximum, 609 |
| IncludeReference, 531 | OriginMinimum, 609 |
| Inverted, 531 | TrailingScale, 610 |
| NoAttribute, 531 | QwtWidgetOverlay |
| Symmetric, 531 | AlphaMask, 639 |
| QwtScaleWidget | AutoRenderMode, 640 |
| TitleInverted, 542 | CopyAlphaMask, 640 |
| QwtSlider | DrawOverlay, 640 |
| LeadingScale, 564 | MaskHint, 639 |
| | |
| NoScale, 564 | NoMask, 639 |
| TrailingScale, 564 | QwtAbstractLegend, 32 |
| QwtSpline | isEmpty, 33 |
| Natural, 572 | QwtAbstractLegend, 33 |
| Periodic, 572 | _ |
| | QwtAbstractLegend, 33 |
| QwtSplineCurveFitter | QwtAbstractLegend, 33 renderLegend, 33 |
| | QwtAbstractLegend, 33 |

| undatal around 24 | Out Abatract Clidar 50 |
|--|-------------------------|
| updateLegend, 34 | QwtAbstractSlider, 52 |
| QwtAbstractScale, 34 abstractScaleDraw, 37 | incrementValue, 54 |
| • | incrementedValue, 54 |
| invTransform, 37 | invertedControls, 54 |
| isInverted, 37 | isReadOnly, 55 |
| lowerBound, 37 | isScrollPosition, 55 |
| maximum, 37 | isTracking, 55 |
| minimum, 38 | isValid, 55 |
| QwtAbstractScale, 36 | keyPressEvent, 55 |
| QwtAbstractScale, 36 | mouseMoveEvent, 56 |
| rescale, 38 | mousePressEvent, 56 |
| scaleDiv, 38 | mouseReleaseEvent, 56 |
| scaleEngine, 38 | pageSteps, 56 |
| scaleMap, 38 | QwtAbstractSlider, 54 |
| scaleMaxMajor, 39 | QwtAbstractSlider, 54 |
| scaleMaxMinor, 39 | scaleChange, 57 |
| scaleStepSize, 39 | scrolledTo, 57 |
| setAbstractScaleDraw, 39 | setInvertedControls, 57 |
| setLowerBound, 39 | setPageSteps, 57 |
| setScale, 39, 40 | setReadOnly, 58 |
| setScaleEngine, 40 | setSingleSteps, 58 |
| setScaleMaxMajor, 40 | setStepAlignment, 58 |
| setScaleMaxMinor, 41 | setTotalSteps, 58 |
| setScaleStepSize, 41 | setTracking, 59 |
| setUpperBound, 41 | setValid, 59 |
| transform, 41 | setValue, 59 |
| upperBound, 42 | setWrapping, 59 |
| QwtAbstractScaleDraw, 42 | singleSteps, 59 |
| draw, 44 | sliderMoved, 60 |
| drawBackbone, 44 | sliderPressed, 60 |
| drawLabel, 44 | sliderReleased, 60 |
| drawTick, 44 | stepAlignment, 60 |
| enableComponent, 46 | totalSteps, 60 |
| extent, 46 | valueChanged, 60 |
| hasComponent, 46 | wheelEvent, 61 |
| invalidateCache, 46 | wrapping, 61 |
| label, 47 | QwtAlphaColorMap, 61 |
| maxTickLength, 47 | color, 62 |
| minimumExtent, 47 | QwtAlphaColorMap, 62 |
| penWidth, 47 | QwtAlphaColorMap, 62 |
| QwtAbstractScaleDraw, 44 | rgb, <mark>62</mark> |
| QwtAbstractScaleDraw, 44 | setColor, 62 |
| ScaleComponent, 43 | QwtAnalogClock, 63 |
| scaleDiv, 47 | drawHand, 66 |
| scaleMap, 47, 48 | drawNeedle, 66 |
| setMinimumExtent, 48 | Hand, 64 |
| setPenWidth, 48 | hand, 66 |
| setScaleDiv, 48 | QwtAnalogClock, 65 |
| setSpacing, 48 | QwtAnalogClock, 65 |
| setTickLength, 48 | setHand, 67 |
| setTransformation, 50 | setTime, 67 |
| spacing, 50 | QwtArraySeriesData |
| tickLabel, 50 | QwtArraySeriesData, 68 |
| tickLength, 50 | QwtArraySeriesData, 68 |
| QwtAbstractSeriesStore, 50 | sample, 68 |
| dataRect, 51 | samples, 68 |
| dataSize, 51 | setSamples, 68 |
| setRectOfInterest, 51 | size, 69 |
| | |

| QwtArraySeriesData< T >, 67 | Style, 83 |
|-----------------------------|-------------------------|
| QwtArrowButton, 69 | QwtCompassRose, 84 |
| arrowSize, 70 | draw, 84 |
| drawArrow, 70 | palette, 85 |
| drawButtonLabel, 70 | QwtCompassScaleDraw, 85 |
| labelRect, 71 | label, 86 |
| paintEvent, 71 | labelMap, 86 |
| QwtArrowButton, 70 | QwtCompassScaleDraw, 86 |
| QwtArrowButton, 70 | QwtCompassScaleDraw, 86 |
| sizeHint, 71 | setLabelMap, 86 |
| QwtCPointerData, 97 | QwtCompassWindArrow, 87 |
| boundingRect, 98 | drawNeedle, 88 |
| QwtCPointerData, 97 | QwtCompassWindArrow, 88 |
| QwtCPointerData, 97 | QwtCompassWindArrow, 88 |
| sample, 98 | Style, 88 |
| size, 98 | QwtCounter, 88 |
| xData, 98 | Button, 90 |
| yData, 98 | buttonReleased, 91 |
| QwtClipper, 71 | event, 91 |
| clipCircle, 71 | incSteps, 91 |
| clipPolygon, 72 | isReadOnly, 91 |
| clipPolygonF, 72 | isValid, 92 |
| | |
| QwtColorMap, 72 | keyPressEvent, 92 |
| color, 74 | maximum, 92 |
| colorIndex, 74 | minimum, 92 |
| colorTable, 74 | numButtons, 93 |
| Format, 73 | QwtCounter, 91 |
| format, 74 | QwtCounter, 91 |
| rgb, 75 | setIncSteps, 93 |
| QwtColumnRect, 75 | setMaximum, 93 |
| Direction, 76 | setMinimum, 93 |
| orientation, 76 | setNumButtons, 93 |
| toRect, 76 | setRange, 94 |
| QwtColumnSymbol, 76 | setReadOnly, 94 |
| draw, 77 | setSingleStep, 94 |
| drawBox, 78 | setStepButton1, 94 |
| FrameStyle, 77 | setStepButton2, 94 |
| frameStyle, 78 | setStepButton3, 95 |
| lineWidth, 78 | setValid, 95 |
| palette, 78 | setValue, 95 |
| QwtColumnSymbol, 77 | setWrapping, 95 |
| QwtColumnSymbol, 77 | singleStep, 95 |
| setFrameStyle, 78 | value, 96 |
| setLineWidth, 79 | valueChanged, 96 |
| setPalette, 79 | wheelEvent, 96 |
| setStyle, 79 | wrapping, 96 |
| Style, 77 | QwtCurveFitter, 99 |
| style, 79 | fitCurve, 99 |
| QwtCompass, 79 | QwtDate, 100 |
| drawRose, 81 | ceil, 101 |
| drawScaleContents, 81 | dateOfWeek0, 101 |
| keyPressEvent, 81 | floor, 102 |
| QwtCompass, 81 | IntervalType, 100 |
| QwtCompass, 81 | maxDate, 102 |
| rose, 81, 82 | minDate, 102 |
| setRose, 82 | toDateTime, 103 |
| QwtCompassMagnetNeedle, 82 | toDouble, 103 |
| | |
| drawNeedle, 83 | toString, 103 |

| utcOffset, 104 | scaleChange, 121 |
|---------------------------------------|---|
| Week0Type, 101 | scaleDraw, 121 |
| weekNumber, 104 | scaleInnerRect, 121 |
| QwtDateScaleDraw, 105 | scrolledTo, 122 |
| dateFormat, 107 | setFrameShadow, 122 |
| dateFormatOfDate, 107 | setLineWidth, 122 |
| intervalType, 107 | setMaxScaleArc, 122 |
| label, 107 | setMinScaleArc, 123 |
| QwtDateScaleDraw, 106 | setMode, 123 |
| QwtDateScaleDraw, 106 | setNeedle, 123 |
| setDateFormat, 108 | setOrigin, 123 |
| setTimeSpec, 108 | setScaleArc, 123 |
| setUtcOffset, 108 | setScaleDraw, 124 |
| setWeek0Type, 108 | Shadow, 117 |
| timeSpec, 109 | sizeHint, 124 |
| toDateTime, 109 | wheelEvent, 124 |
| utcOffset, 109 | QwtDialNeedle, 124 |
| week0Type, 109 | draw, 125 |
| QwtDateScaleEngine, 110 | drawNeedle, 125 |
| alignDate, 111 | palette, 126 |
| autoScale, 111 | setPalette, 126 |
| divideScale, 112 | QwtDialSimpleNeedle, 126 drawNeedle, 127 |
| intervalType, 112 | QwtDialSimpleNeedle, 127 |
| maxWeeks, 112 QwtDateScaleEngine, 111 | QwtDialSimpleNeedle, 127 |
| QwtDateScaleEngine, 111 | setWidth, 128 |
| setMaxWeeks, 113 | Style, 127 |
| setTimeSpec, 113 | width, 128 |
| setUtcOffset, 113 | QwtDynGridLayout, 128 |
| setWeek0Type, 113 | addItem, 130 |
| timeSpec, 114 | columnsForWidth, 130 |
| toDateTime, 114 | count, 130 |
| utcOffset, 114 | expandingDirections, 130 |
| week0Type, 114 | hasHeightForWidth, 130 |
| QwtDial, 115 | heightForWidth, 130 |
| boundingRect, 117 | isEmpty, 131 |
| changeEvent, 118 | itemAt, 131 |
| drawContents, 118 | itemCount, 131 |
| drawFocusIndicator, 118 | layoutGrid, 131 |
| drawFrame, 118 | layoutItems, 131 |
| drawNeedle, 118 | maxColumns, 132 |
| drawScale, 119 | maxItemWidth, 132 |
| drawScaleContents, 119 | numColumns, 132 |
| frameShadow, 119 | numRows, 132 |
| innerRect, 119 | QwtDynGridLayout, 129 |
| invalidateCache, 119 | QwtDynGridLayout, 129 |
| isScrollPosition, 119 | setExpandingDirections, 132 |
| lineWidth, 120 | setGeometry, 133 |
| maxScaleArc, 120 | setMaxColumns, 133 |
| minScaleArc, 120 | sizeHint, 133 |
| minimumSizeHint, 120 | stretchGrid, 133 |
| Mode, 117 | takeAt, 133 |
| mode, 120 | QwtEventPattern, 134 |
| needle, 120, 121 | initKeyPattern, 137 |
| origin, 121 | initMousePattern, 137 |
| paintEvent, 121 | keyMatch, 137 |
| QwtDial, 117 | keyPattern, 138 |
| QwtDial, 117 | KeyPatternCode, 135 |
| | |

| mouseMatch, 138 | setMinValue, 159 |
|-----------------------------------|----------------------------|
| mousePattern, 139 | symmetrize, 160 |
| MousePatternCode, 136 | width, 160 |
| QwtEventPattern, 137 | QwtIntervalSample, 160 |
| QwtEventPattern, 137 | QwtIntervalSample, 161 |
| setKeyPattern, 139 | QwtIntervalSample, 161 |
| setMousePattern, 139 | QwtIntervalSeriesData, 161 |
| QwtEventPattern::KeyPattern, 31 | boundingRect, 163 |
| QwtEventPattern::MousePattern, 31 | QwtIntervalSeriesData, 161 |
| QwtGraphic, 139 | QwtIntervalSeriesData, 161 |
| boundingRect, 144 | QwtIntervalSymbol, 163 |
| commands, 144 | brush, 164 |
| controlPointRect, 144 | draw, 164 |
| defaultSize, 144 | pen, 165 |
| drawlmage, 144 | QwtIntervalSymbol, 164 |
| drawPath, 146 | QwtIntervalSymbol, 164 |
| drawPixmap, 146 | setBrush, 165 |
| isEmpty, 146 | setPen, 165 |
| isNull, 146 | setStyle, 166 |
| operator=, 146 | setWidth, 166 |
| QwtGraphic, 142 | Style, 164 |
| QwtGraphic, 142 | style, 166 |
| render, 148 | width, 166 |
| RenderHint, 142 | QwtKnob, 166 |
| RenderHints, 142 | alignment, 169 |
| reset, 148 | changeEvent, 170 |
| scaledBoundingRect, 149 | drawFocusIndicator, 170 |
| setCommands, 149 | drawKnob, 170 |
| setDefaultSize, 149 | drawMarker, 170 |
| setRenderHint, 149 | isScrollPosition, 170 |
| sizeMetrics, 150 | knobRect, 170 |
| testRenderHint, 150 | KnobStyle, 169 |
| tolmage, 150 | knobStyle, 171 |
| toPixmap, 151 | markerSize, 171 |
| updateState, 151 | MarkerStyle, 169 |
| QwtInterval, 152 | markerStyle, 171 |
| BorderFlag, 153 | minimumSizeHint, 171 |
| borderFlags, 154 | numTurns, 171 |
| contains, 154 | paintEvent, 171 |
| extend, 154 | QwtKnob, 169 |
| intersect, 154 | QwtKnob, 169 |
| intersects, 155 | scaleDraw, 173 |
| invalidate, 155 | scrolledTo, 173 |
| inverted, 155 | setAlignment, 173 |
| isNull, 155 | setBorderWidth, 173 |
| isValid, 155 | setKnobStyle, 174 |
| limited, 155 | setKnobWidth, 174 |
| maxValue, 156 | setMarkerSize, 174 |
| minValue, 156 | setMarkerStyle, 174 |
| normalized, 156 | setNumTurns, 174 |
| operator==, 158 | setScaleDraw, 175 |
| operator&, 156 | setTotalAngle, 175 |
| operator&=, 156 | sizeHint, 175 |
| QwtInterval, 153 | totalAngle, 175 |
| QwtInterval, 153 | QwtLegend, 176 |
| setBorderFlags, 159 | checked, 177 |
| setInterval, 159 | clicked, 178 |
| setMaxValue, 159 | contentsWidget, 178 |
| | 22 |

| createWidget, 178 | setMode, 193 |
|--------------------------|------------------------------|
| defaultItemMode, 179 | QwtLinearScaleEngine, 193 |
| eventFilter, 179 | align, 194 |
| heightForWidth, 179 | autoScale, 194 |
| horizontalScrollBar, 179 | buildMajorTicks, 195 |
| isEmpty, 179 | buildMinorTicks, 195 |
| itemChecked, 179 | buildTicks, 195 |
| itemClicked, 180 | divideScale, 195 |
| itemInfo, 180 | QwtLinearScaleEngine, 194 |
| legendWidget, 180 | QwtLinearScaleEngine, 194 |
| legendWidgets, 180 | QwtLogScaleEngine, 196 |
| maxColumns, 180 | align, 197 |
| QwtLegend, 177 | autoScale, 197 |
| QwtLegend, 177 | buildMajorTicks, 198 |
| renderItem, 181 | buildMinorTicks, 198 |
| renderLegend, 181 | buildTicks, 198 |
| scrollExtent, 181 | divideScale, 198 |
| setDefaultItemMode, 181 | QwtLogScaleEngine, 197 |
| setMaxColumns, 182 | QwtLogScaleEngine, 197 |
| updateLegend, 182 | QwtLogTransform, 199 |
| updateWidget, 182 | bounded, 200 |
| verticalScrollBar, 182 | copy, 200 |
| QwtLegendData, 183 | invTransform, 200 |
| hasRole, 184 | transform, 200 |
| icon, 184 | QwtMagnifier, 200 |
| isValid, 184 | eventFilter, 202 |
| Mode, 183 | getMouseButton, 202 |
| mode, 184 | getZoomInKey, 202 |
| setValue, 184 | getZoomOutKey, 203 |
| setValues, 184 | isEnabled, 203 |
| title, 184 | keyFactor, 203 |
| value, 184 | mouseFactor, 203 |
| values, 186 | parentWidget, 203 |
| QwtLegendLabel, 186 | QwtMagnifier, 202 |
| data, 188 | QwtMagnifier, 202 |
| icon, 188 | rescale, 204 |
| itemMode, 188 | setEnabled, 205 |
| QwtLegendLabel, 187 | setKeyFactor, 205 |
| QwtLegendLabel, 187 | setMouseButton, 205 |
| setChecked, 188 | setMouseFactor, 205 |
| setData, 188 | setWheelFactor, 205 |
| setlcon, 188 | setWheelModifiers, 206 |
| setItemMode, 189 | setZoomInKey, 206 |
| setSpacing, 189 | setZoomOutKey, 206 |
| setText, 189 | wheelFactor, 206 |
| spacing, 189 | wheelModifiers, 207 |
| QwtLinearColorMap, 189 | widgetKeyPressEvent, 207 |
| addColorStop, 191 | widgetKeyReleaseEvent, 207 |
| color1, 191 | widgetMouseMoveEvent, 207 |
| color2, 191 | widgetMousePressEvent, 207 |
| colorIndex, 192 | widgetMouseReleaseEvent, 208 |
| colorStops, 192 | widgetWheelEvent, 208 |
| Mode, 190 | QwtMathMLTextEngine, 208 |
| mode, 192 | draw, 209 |
| QwtLinearColorMap, 191 | heightForWidth, 209 |
| QwtLinearColorMap, 191 | mightRender, 210 |
| rgb, 192 | textMargins, 210 |
| setColorInterval, 192 | textSize, 210 |
| Solodio intol val, 132 | IUAIUIZG, Z IU |

| QwtMatrixRasterData, 210 | isEnabled, 231 |
|-----------------------------|------------------------------|
| numColumns, 212 | isOrientationEnabled, 231 |
| numRows, 212 | moved, 231 |
| pixelHint, 212 | paintEvent, 232 |
| ResampleMode, 212 | panned, 232 |
| resampleMode, 212 | QwtPanner, 230 |
| setInterval, 213 | QwtPanner, 230 |
| setResampleMode, 213 | setAbortKey, 232 |
| , | • |
| setValue, 213 | setCursor, 232 |
| setValueMatrix, 213 | setEnabled, 232 |
| value, 214 | setMouseButton, 232 |
| valueMatrix, 214 | setOrientations, 232 |
| QwtNullPaintDevice, 214 | widgetKeyPressEvent, 233 |
| metric, 216 | widgetKeyReleaseEvent, 233 |
| Mode, 216 | widgetMouseMoveEvent, 233 |
| mode, 216 | widgetMousePressEvent, 233 |
| setMode, 217 | widgetMouseReleaseEvent, 233 |
| sizeMetrics, 217 | QwtPicker, 234 |
| QwtNullTransform, 217 | accept, 239 |
| copy, 218 | activated, 239 |
| invTransform, 218 | adjustedPoints, 239 |
| transform, 218 | append, 240 |
| QwtOHLCSample, 218 | appended, 240 |
| boundingInterval, 219 | begin, 240 |
| isValid, 219 | changed, 240 |
| QwtOHLCSample, 219 | DisplayMode, 237 |
| • | |
| QwtOHLCSample, 219 | drawRubberBand, 240 |
| time, 220 | drawTracker, 241 |
| QwtPainter, 220 | end, 241 |
| backingStore, 222 | eventFilter, 241 |
| drawBackgound, 222 | isActive, 241 |
| drawColorBar, 222 | isEnabled, 242 |
| drawFrame, 222 | move, 242 |
| drawRoundFrame, 223 | moved, 242 |
| drawRoundedFrame, 223 | pickArea, 242 |
| drawSimpleRichText, 223 | pickedPoints, 242 |
| fillPixmap, 223 | QwtPicker, 238 |
| isAligning, 224 | QwtPicker, 238 |
| isX11GraphicsSystem, 224 | remove, 242 |
| polylineSplitting, 224 | removed, 243 |
| roundingAlignment, 224 | reset. 243 |
| setPolylineSplitting, 225 | ResizeMode, 237 |
| setRoundingAlignment, 225 | resizeMode, 243 |
| QwtPainterCommand, 225 | RubberBand, 238 |
| imageData, 227 | rubberBand, 243 |
| - | rubberBandMask, 243 |
| operator=, 227 | • |
| path, 228 | rubberBandOverlay, 243 |
| pixmapData, 228 | rubberBandPen, 243 |
| QwtPainterCommand, 226, 227 | selected, 244 |
| QwtPainterCommand, 226, 227 | selection, 244 |
| stateData, 228 | setEnabled, 244 |
| Type, 226 | setResizeMode, 244 |
| type, 228 | setRubberBand, 244 |
| QwtPanner, 228 | setRubberBandPen, 245 |
| contentsMask, 230 | setStateMachine, 245 |
| cursor, 230 | setTrackerFont, 245 |
| eventFilter, 231 | setTrackerMode, 245 |
| grab, 231 | setTrackerPen, 245 |
| 9 , - | |

| eteteMechine 247 | aviaWidget 071 |
|--|--------------------------------|
| stateMachine, 247 stretchSelection, 247 | axisWidget, 271 canvas, 272 |
| trackerFont, 247 | canvasBackground, 272 |
| trackerMode, 247 | canvasMap, 272 |
| tracker/Node, 247 | drawCanvas, 272 |
| trackerOverlay, 247 | drawltems, 272 |
| trackerPosition, 248 | enableAxis, 274 |
| tracker osition, 248 | event, 274 |
| trackerTect, 248 | eventFilter, 274 |
| transition, 248 | footer, 274 |
| widgetEnterEvent, 249 | footerLabel, 275 |
| widgetKeyPressEvent, 249 | getCanvasMarginsHint, 275 |
| widgetKeyReleaseEvent, 249 | grabProperties, 275 |
| widgetLeaveEvent, 249 | infoToItem, 275 |
| widgetMouseDoubleClickEvent, 249 | insertLegend, 276 |
| widgetMouseMoveEvent, 251 | invTransform, 276 |
| widgetMousePressEvent, 251 | itemAttached, 276 |
| widgetMouseReleaseEvent, 251 | itemToInfo, 277 |
| widgetWheelEvent, 251 | legend, 277 |
| QwtPickerClickPointMachine, 252 | legendDataChanged, 277 |
| QwtPickerClickRectMachine, 252 | LegendPosition, 267 |
| QwtPickerDragLineMachine, 254 | plotLayout, 278 |
| QwtPickerDragPointMachine, 254 | QwtPlot, 268 |
| QwtPickerDragRectMachine, 255 | QwtPlot, 268 |
| QwtPickerMachine, 256 | replot, 278 |
| SelectionType, 257 | resizeEvent, 278 |
| QwtPickerPolygonMachine, 258 | setAutoReplot, 278 |
| QwtPickerTrackerMachine, 259 | setAxisAutoScale, 278 |
| QwtPixelMatrix, 259 | setAxisFont, 279 |
| index, 260 | setAxisLabelAlignment, 279 |
| QwtPixelMatrix, 260 | setAxisLabelRotation, 279 |
| QwtPixelMatrix, 260 | setAxisMaxMajor, 279 |
| rect, 261 | setAxisMaxMinor, 280 |
| setRect, 261 | setAxisScale, 280 |
| testAndSetPixel, 261 | setAxisScaleDiv, 280 |
| testPixel, 261 | setAxisScaleDraw, 280 |
| QwtPlainTextEngine, 261 | setAxisScaleEngine, 281 |
| draw, 262 | setAxisTitle, 281 |
| heightForWidth, 263 | setCanvas, 281 |
| mightRender, 263 | setCanvasBackground, 282 |
| textMargins, 263 | setFooter, 282 |
| textSize, 263 | setPlotLayout, 282 |
| QwtPlot, 264 | setTitle, 282, 283 |
| applyProperties, 268 | sizeHint, 283 |
| autoReplot, 268 | title, 283 |
| Axis, 267 | titleLabel, 283 |
| axisAutoScale, 268 | transform, 283 |
| axisEnabled, 269 | updateAxes, 283 |
| axisFont, 269 | updateCanvasMargins, 284 |
| axisInterval, 269 | updateLayout, 284 |
| axisMaxMajor, 269 | updateLegend, 284 |
| axisMaxMinor, 269 | QwtPlotAbstractBarChart, 285 |
| axisScaleDiv, 270 | baseline, 286 |
| axisScaleDraw, 270 | getCanvasMarginHint, 287 |
| axisScaleEngine, 270 | layoutHint, 287 |
| axisStepSize, 271 | LayoutPolicy, 286 |
| axisTitle, 271 | layoutPolicy, 287 |
| axisValid, 271 | margin, 287 |
| • | <i>5 /</i> |

| QwtPlotAbstractBarChart, 286 | drawSteps, 309 |
|------------------------------|---------------------------|
| QwtPlotAbstractBarChart, 286 | drawSticks, 309 |
| sampleWidth, 288 | drawSymbols, 310 |
| setBaseline, 288 | fillCurve, 310 |
| setLayoutHint, 288 | LegendAttribute, 305 |
| setLayoutPolicy, 288 | legendlcon, 310 |
| setMargin, 289 | PaintAttribute, 305 |
| setSpacing, 289 | pen, 311 |
| spacing, 289 | QwtPlotCurve, 306, 307 |
| QwtPlotBarChart, 290 | QwtPlotCurve, 306, 307 |
| barTitle, 292 | rtti, 311 |
| boundingRect, 292 | setBaseline, 311 |
| drawBar, 292 | setBrush, 311 |
| drawSample, 293 | setCurveAttribute, 312 |
| drawSeries, 293 | setCurveFitter, 312 |
| legendData, 293 | setLegendAttribute, 312 |
| legendlcon, 293 | setPaintAttribute, 312 |
| LegendMode, 291 | setPen, 312, 313 |
| legendMode, 294 | setRawSamples, 313 |
| QwtPlotBarChart, 292 | setSamples, 313, 314 |
| QwtPlotBarChart, 292 | setStyle, 314 |
| rtti, 294 | setSymbol, 314 |
| setLegendMode, 294 | style, 315 |
| setSamples, 294, 295 | symbol, 315 |
| setSymbol, 295 | testCurveAttribute, 315 |
| specialSymbol, 295 | testLegendAttribute, 315 |
| symbol, 296 | testPaintAttribute, 315 |
| QwtPlotCanvas, 296 | QwtPlotDict, 316 |
| backingStore, 299 | ∼QwtPlotDict, 317 |
| borderPath, 299 | autoDelete, 317 |
| borderRadius, 299 | detachltems, 317 |
| drawBorder, 299 | insertItem, 317 |
| drawFocusIndicator, 299 | itemList, 317 |
| event, 300 | QwtPlotDict, 316 |
| FocusIndicator, 297 | QwtPlotDict, 316 |
| focusIndicator, 300 | removeltem, 318 |
| PaintAttribute, 298 | setAutoDelete, 318 |
| paintEvent, 300 | QwtPlotDirectPainter, 318 |
| QwtPlotCanvas, 298 | Attribute, 319 |
| QwtPlotCanvas, 298 | clipRegion, 320 |
| replot, 300 | drawSeries, 320 |
| resizeEvent, 300 | hasClipping, 320 |
| setBorderRadius, 300 | setAttribute, 320 |
| setFocusIndicator, 301 | setClipRegion, 321 |
| setPaintAttribute, 301 | setClipping, 321 |
| testPaintAttribute, 301 | testAttribute, 321 |
| QwtPlotCurve, 302 | QwtPlotGLCanvas, 321 |
| baseline, 307 | borderPath, 324 |
| brush, 307 | drawBackground, 324 |
| closePolyline, 307 | drawBorder, 324 |
| closest bryline, 307 | drawltems, 324 |
| CurveAttribute, 304 | event, 324 |
| curveFitter, 308 | frameRect, 324 |
| CurveStyle, 304 | frameShadow, 324 |
| drawCurve, 308 | frameShape, 325 |
| drawDots, 308 | frameStyle, 325 |
| drawLines, 308 | frameWidth, 325 |
| drawSeries, 309 | lineWidth, 325 |
| ulawoelies, ous | iiilevviu(ii, 323 |

| midLineWidth, 325 | brush, 343 |
|---|---------------------------|
| paintEvent, 325 | CurveStyle, 342 |
| QwtPlotGLCanvas, 323 | drawSeries, 344 |
| QwtPlotGLCanvas, 323 | drawSymbols, 344 |
| setFrameShadow, 326 | drawTube, 344 |
| setFrameShape, 326 | legendlcon, 345 |
| setFrameStyle, 326 | PaintAttribute, 343 |
| setLineWidth, 326 | |
| , in the second | pen, 345 |
| setMidLineWidth, 326 | QwtPlotIntervalCurve, 343 |
| Shadow, 323 | QwtPlotIntervalCurve, 343 |
| Shape, 323 | rtti, 345 |
| QwtPlotGrid, 327 | setBrush, 345 |
| draw, 328 | setPaintAttribute, 345 |
| enableX, 328 | setPen, 346 |
| enableXMin, 328 | setSamples, 346 |
| enableY, 329 | setStyle, 346 |
| enableYMin, 329 | setSymbol, 348 |
| majorPen, 329 | style, 348 |
| minorPen, 329 | symbol, 348 |
| rtti, 329 | testPaintAttribute, 348 |
| setMajorPen, 329, 330 | QwtPlotItem, 348 |
| setMinorPen, 330 | attach, 353 |
| setPen, 330, 331 | boundingRect, 353 |
| setXDiv, 331 | defaultIcon, 353 |
| setYDiv, 331 | detach, 354 |
| updateScaleDiv, 331 | draw, 354 |
| · | • |
| xEnabled, 331 | getCanvasMarginHint, 354 |
| xMinEnabled, 332 | isVisible, 354 |
| xScaleDiv, 332 | ItemAttribute, 351 |
| yEnabled, 332 | itemChanged, 355 |
| yMinEnabled, 332 | ItemInterest, 352 |
| yScaleDiv, 332 | legendChanged, 355 |
| QwtPlotHistogram, 332 | legendData, 355 |
| baseline, 335 | legendlcon, 355 |
| boundingRect, 335 | legendlconSize, 356 |
| brush, 335 | paintRect, 356 |
| columnRect, 335 | QwtPlotItem, 353 |
| drawColumn, 335 | QwtPlotItem, 353 |
| drawColumns, 337 | RenderHint, 352 |
| drawLines, 337 | renderThreadCount, 356 |
| drawOutline, 337 | rtti, 356 |
| drawSeries, 338 | RttiValues, 352 |
| HistogramStyle, 334 | scaleRect, 356 |
| legendicon, 338 | setAxes, 358 |
| pen, 338 | setItemAttribute, 358 |
| QwtPlotHistogram, 334, 335 | setItemInterest, 358 |
| QwtPlotHistogram, 334, 335 | setLegendIconSize, 358 |
| rtti, 338 | setRenderHint, 359 |
| | |
| setBaseline, 339 | setRenderThreadCount, 359 |
| setBrush, 339 | setTitle, 359 |
| setPen, 339 | setVisible, 359 |
| setSamples, 340 | setXAxis, 360 |
| setStyle, 340 | setYAxis, 360 |
| setSymbol, 340 | setZ, 360 |
| style, 340 | testItemAttribute, 360 |
| symbol, 341 | testItemInterest, 360 |
| QwtPlotIntervalCurve, 341 | testRenderHint, 362 |
| boundingRect, 343 | title, 362 |
| | |

| updateLegend, 362 | setBorderDistance, 379 |
|-----------------------------|---------------------------|
| updateScaleDiv, 362 | setBorderPen, 379 |
| z, 363 | setBorderRadius, 379 |
| QwtPlotLayout, 363 | setFont, 379 |
| activate, 365 | setItemMargin, 380 |
| alignCanvasToScale, 365 | setItemSpacing, 380 |
| alignLegend, 365 | setMargin, 380 |
| alignScales, 366 | setMaxColumns, 380 |
| canvasMargin, 366 | setSpacing, 380 |
| canvasRect, 366 | setTextPen, 381 |
| expandLineBreaks, 366 | spacing, 381 |
| footerRect, 367 | textPen, 381 |
| invalidate, 367 | updateLegend, 381 |
| layoutLegend, 367 | QwtPlotMagnifier, 381 |
| legendPosition, 367 | isAxisEnabled, 383 |
| legendRatio, 367 | QwtPlotMagnifier, 383 |
| legendRect, 368 | QwtPlotMagnifier, 383 |
| - | _ |
| minimumSizeHint, 368 | rescale, 383 |
| Option, 364 | setAxisEnabled, 383 |
| scaleRect, 368 | QwtPlotMarker, 383 |
| setAlignCanvasToScale, 368 | boundingRect, 386 |
| setAlignCanvasToScales, 369 | draw, 386 |
| setCanvasMargin, 369 | drawLabel, 386 |
| setCanvasRect, 369 | drawLines, 386 |
| setFooterRect, 369 | label, 386 |
| setLegendPosition, 369, 371 | labelAlignment, 387 |
| setLegendRatio, 371 | labelOrientation, 387 |
| setLegendRect, 371 | legendlcon, 387 |
| setScaleRect, 371 | linePen, 387 |
| setSpacing, 372 | LineStyle, 385 |
| setTitleRect, 372 | lineStyle, 387 |
| spacing, 372 | rtti, 387 |
| titleRect, 372 | setLabel, 388 |
| QwtPlotLegendItem, 372 | setLabelAlignment, 388 |
| alignment, 375 | setLabelOrientation, 388 |
| backgroundBrush, 375 | setLinePen, 388, 389 |
| BackgroundMode, 374 | setLineStyle, 389 |
| backgroundMode, 375 | setSpacing, 389 |
| borderDistance, 375 | setSymbol, 389 |
| borderPen, 375 | spacing, 389 |
| borderRadius, 375 | symbol, 390 |
| draw, 376 | QwtPlotMultiBarChart, 390 |
| drawBackground, 376 | barTitles, 392 |
| drawLegendData, 376 | boundingRect, 392 |
| font, 376 | ChartStyle, 392 |
| geometry, 376 | drawBar, 392 |
| heightForWidth, 377 | drawGroupedBars, 393 |
| itemMargin, 377 | drawSample, 393 |
| itemSpacing, 377 | drawSarriple, 393 |
| , - | |
| legendGeometries, 377 | drawStackedBars, 394 |
| margin, 377 | legendData, 394 |
| maxColumns, 377 | legendlcon, 394 |
| minimumSize, 378 | QwtPlotMultiBarChart, 392 |
| plotItems, 378 | QwtPlotMultiBarChart, 392 |
| rtti, 378 | resetSymbolMap, 395 |
| setAlignment, 378 | rtti, 395 |
| setBackgroundBrush, 378 | setBarTitles, 395 |
| setBackgroundMode, 379 | setSamples, 395 |
| | |

| setStyle, 396 | renderLegend, 416 |
|-------------------------|----------------------------|
| setSymbol, 396 | renderScale, 416 |
| specialSymbol, 396 | renderTitle, 417 |
| style, 396 | renderTo, 417 |
| symbol, 397 | setDiscardFlag, 417 |
| QwtPlotPanner, 398 | setDiscardFlags, 417 |
| contentsMask, 399 | setLayoutFlag, 419 |
| grab, 399 | setLayoutFlags, 419 |
| isAxisEnabled, 400 | testDiscardFlag, 419 |
| moveCanvas, 400 | testLayoutFlag, 419 |
| QwtPlotPanner, 399 | QwtPlotRescaler, 420 |
| QwtPlotPanner, 399 | aspectRatio, 423 |
| setAxisEnabled, 400 | canvas, 423 |
| QwtPlotPicker, 400 | canvasResizeEvent, 423 |
| | |
| append, 403 | expandInterval, 424 |
| appended, 403 | expandScale, 424 |
| canvas, 403 | ExpandingDirection, 421 |
| end, 403 | expandingDirection, 423 |
| invTransform, 404 | interval, 424 |
| move, 404 | intervalHint, 424 |
| moved, 404 | isEnabled, 424 |
| plot, 405 | orientation, 425 |
| QwtPlotPicker, 402 | plot, 425 |
| QwtPlotPicker, 402 | QwtPlotRescaler, 422 |
| scaleRect, 405 | QwtPlotRescaler, 422 |
| selected, 405 | referenceAxis, 425 |
| setAxis, 405 | rescale, 425 |
| trackerText, 406 | RescalePolicy, 421 |
| trackerTextF, 406 | rescalePolicy, 425 |
| transform, 406 | setAspectRatio, 425, 427 |
| QwtPlotRasterItem, 407 | setEnabled, 427 |
| alpha, 409 | setExpandingDirection, 427 |
| boundingRect, 409 | setIntervalHint, 427 |
| CachePolicy, 408 | setReferenceAxis, 429 |
| cachePolicy, 409 | setRescalePolicy, 429 |
| draw, 409 | syncScale, 429 |
| imageMap, 409 | updateScales, 429 |
| - · | • |
| interval, 410 | QwtPlotScaleItem, 429 |
| invalidateCache, 410 | borderDistance, 431 |
| PaintAttribute, 408 | font, 431 |
| pixelHint, 410 | isScaleDivFromAxis, 431 |
| renderImage, 410 | palette, 432 |
| setAlpha, 411 | position, 432 |
| setCachePolicy, 411 | QwtPlotScaleItem, 431 |
| setPaintAttribute, 411 | QwtPlotScaleItem, 431 |
| testPaintAttribute, 412 | rtti, 432 |
| QwtPlotRenderer, 412 | scaleDiv, 432 |
| DiscardFlag, 413 | scaleDraw, 432 |
| discardFlags, 414 | setAlignment, 432 |
| exportTo, 414 | setBorderDistance, 433 |
| LayoutFlag, 413 | setFont, 433 |
| layoutFlags, 414 | setPalette, 433 |
| QwtPlotRenderer, 414 | setPosition, 433 |
| QwtPlotRenderer, 414 | setScaleDiv, 434 |
| render, 415 | setScaleDivFromAxis, 434 |
| renderCanvas, 415 | setScaleDraw, 434 |
| renderDocument, 415 | updateScaleDiv, 434 |
| renderFooter, 416 | QwtPlotSeriesItem, 435 |
| Totaldit ootor, 410 | with lotochication, 700 |

| boundingRect, 436 | | | |
|--|------|-------------------------|-------------------------------|
| drawSeries, 436 orientation, 437 CwrPlotSeriestlem, 436 CwrPlotSeriestlem, 436 CwrPlotSeriestlem, 436 CwrPlotSeriestlem, 437 publicSeriestlem, 438 plegendMode, 440 plegendMode, 440 plegendMode, 439 plegendMode, 439 pen, 441 publicSpeltem, 439, 440 penderTolerance, 441 prenderTolerance, 442 prenderTolerance, 443 prenderTolerance, 444 prenderTolerance, 443 prenderTolerance, 443 prenderTolerance, 443 prenderTolerance, 443 prenderTolerance, 444 prenderTolerance, 443 prenderTolerance, 443 prenderTolerance, 443 prenderTolerance, 444 prenderTolerance, 443 prenderTolerance, 443 prenderTolerance, 443 prenderTolerance, 443 prenderTolerance, 444 prenderTolerance, 443 prenderTolerance, 444 prenderTolerance, 445 prenderTolerance, 445 prenderTolerance, 445 prenderTolerance, 446 prenderTolerance, 447 prenderTolerance, 448 prenderTolerance, 449 prenderTolerance, | | boundingRect, 436 | QwtPlotSpectrogram, 450 |
| orientation, 437 OwtPlotSeriesItem, 436 OwtPlotSeriesItem, 436 setOrientation, 437 updateScaleDiv, 437 setConrocrIag, 455 owtPlotShapeItem, 437 brush, 440 legendLoon, 440 LegendMode, 439 legendMode, 439 legendMode, 440 PaintAttribute, 439 ppn, 441 OwtPlotShapeItem, 439, 440 OwtPlotSyltem, 458 loadData, 458 loadData, 458 loadData, 458 loadPile, 459 renderer, 4 | | draw, 436 | QwtPlotSpectrogram, 450 |
| OwtPlotSeriesItem, 436 OwtPlotSeriesItem, 436 OwtPlotSeriesItem, 437 UpdateScaleDiv, 437 UpdateScaleDiv, 437 UpdateScaleDiv, 437 OwtPlotShapeItem, 437 UpdateScaleDiv, 440 UpdateScaleDiv, 445 UpdateScaleDiv, | | drawSeries, 436 | renderContourLines, 453 |
| QwiPlotSeriestlem, 436 setCorlentation, 437 updateScaleDiv, 437 OwtPlotShapeltem, 437 brush, 440 legendlcon, 440 LegendlMode, 439 legendlMode, 440 PaintAttribute, 439 pen, 441 QwiPlotShapeltem, 439, 440 renderTolerance, 441 setBrush, 441 setBrush, 441 setBrush, 442 setPen, 442 setPen, 442 setPen, 442 setPendyon, 442 setRect, 443 setRederTolerance, 443 setShape, 443 shape, 443 stesPaintAttribute, 443 QwiPlotSpetroCurve, 444 colorMap, 445 colorMap, 445 drawSeries, 446 QwiPlotSpectroCurve, 445 qwiPlotSpectroCurve, 446 qwiPlotSpectroCurve, 445 qwiPlotSpectroCurve, 446 qwiPlotSpectroCurve, 445 qwiPlotSpectroCurve, 446 qwiPlotSpectroCurve, 446 qwiPlotSpectroCurve, 445 qwiPlotSpectroCurve, 446 qwiPlotSpectroCurve, 466, 467 qwiPlotS | | orientation, 437 | renderImage, 454 |
| QwiPlotSeriestlem, 436 setCorlentation, 437 updateScaleDiv, 437 SetContown 437 setContown 437 setContown 445 setContown 440 setDisphaphetem, 437 brush, 440 legendlcon, 440 Legendlwode, 439 legendlwode, 439 legendlwode, 441 QwiPlotShapeltem, 439, 440 renderTolerance, 441 setBrush, 441 setBrush, 441 setBrush, 442 setPen, 442 setPen, 442 setPen, 442 setPendrolerance, 443 setRenderTolerance, 443 setRenderTolerance, 443 setShape, 443 shape, 443 testPaintAttribute, 443 QwiPlotSpetroCurve, 444 colorMap, 445 colorMap, 445 drawSeries, 446 QwiPlotSpectroCurve, 445 QwiPlotSpectroCurve, 445 QwiPlotSpectroCurve, 445 qrawSeries, 446 QwiPlotSpectroCurve, 445 qrawSeries, 446 QwiPlotSpectroCurve, 445 qrawSeries, 447 setPenWidth, 446 setColorMap, 447 setPenWidth, 447 setPenWidth, 447 setPenWidth, 447 setPenWidth, 447 setPenWidth, 447 setPenWidth, 447 setPaintAttribute, 448 QwiFlotSpectroCurve, 445 qrawSeries, 447 setPenWidth, 448 QwiFlotSpectroCurve, 445 colorMap, 450 contourLevels, 450 contourLevels, 450 contourLevels, 450 draw, 452 drawContourLines, 452 interval, 453 setSymbolWidth, 469 setMinSymbolWidth, 471 setSymbolExent, 472 setSymbolExent, 472 | | QwtPlotSeriesItem, 436 | renderTile, 454 |
| setColorMap, 454 updateScaleDiv, 437 QwtPlotShapeltem, 437 brush, 440 draw, 440 legendLoon, 440 setDisplayMode, 455 setData, 456 legendLoon, 440 setDisplayMode, 456 legendMode, 439 legendMode, 439 pen, 441 QwtPlotShapeltem, 439, 440 QwtPlotShapeltem, 439, 440 renderTolerance, 441 rtit, 441 setLegendMode, 441 setLegendMode, 441 setPaintAttribute, 442 setPen, 442 setPen, 442 setPen, 442 setPender, 443 setRenderTolerance, 443 setShape, 443 setShape, 443 testPaintAttribute, 443 QwtPlotSrotCurve, 444 colorMap, 445 colorMap, 445 drawDots, 446 drawSeries, 446 PaintAttribute, 445 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 qrawCandata, 447 setPenWidth, 446 setColorMap, 447 setPenWidth, 447 setPenWidth, 447 setPenWidth, 447 setPenWidth, 447 setPenWidth, 447 setPenWidth, 447 setPaintAttribute, 448 QwtPlotSpectroCurve, 445 colorMap, 560 contourLevels, 450 contourLevels, 4 | | | |
| updateScaleDiv, 437 OwtPlotShapeltem, 437 brush, 440 draw, 440 legendIcon, 440 LegendMode, 439 legendMode, 439 legendMode, 441 OwtPlotShapeltem, 439, 440 OwtPlotSygltem, 458 render, 459 renderTolerance, 441 rtit, 441 setBrush, 441 setBrush, 441 setBrush, 442 setPen, 442 setPen, 442 setPolygon, 442 setPolygon, 442 setPolygon, 442 setPolygon, 443 setShape, 443 setShape, 443 setShape, 443 setShape, 443 colorMap, 445 colorMap, 445 drawDots, 446 drawSeries, 446 drawSeries, 446 QwtPlotSpectroCurve, 445 OwtPlotTextLabel, 461 rtit, 462 setMargin, 462 setText, 462 drawDots, 446 drawSeries, 446 QwtPlotSpectroCurve, 445 owtPlotSpectroCurve, 445 owtPlotSpectroCurve, 445 rtit, 446 setColorMap, 447 setPamiAttribute, 447 setPamiAttribute, 447 setPamiAttribute, 447 setPamiAttribute, 448 legendlcon, 488 maxSymbolWidth, 469 minSymbolWidth, 469 minSymbolWidth, 469 minSymbolWidth, 469 setMaxSymbolWidth, 469 setMaxSymbolWidth, 469 setMinSymbolWidth, 471 setSamples, 471 setSymbolExtent, 472 interval, 453 setSymbolPen, 472 | | | |
| QwtPlotShapeltem, 437 setContourLevels, 455 brush, 440 setData, 455 draw, 440 setData, 455 legendlcon, 440 setDisplayMode, 456 Legendlwode, 439 testConrecFlag, 456 legendlwode, 440 testDisplayMode, 456 PaintAttribute, 439 QwtPlotShapeltem, 457 GwtPlotShapeltem, 439, 440 loadData, 458 OwtPlotShapeltem, 439, 440 loadData, 458 renderTolerance, 441 QwtPlotSygltem, 458 renderTolerance, 441 QwtPlotSygltem, 458 render, 459 render, 459 render, 459 re | | | • |
| brush, 440 draw, 440 legendlcon, 440 legendlcon, 440 LegendMode, 439 legendMode, 439 legendMode, 440 PaintAttribute, 439 pen, 441 QwtPlotShapeltem, 439, 440 QwtPlotShapeltem, 439, 440 renderTolerance, 441 rtti, 441 setLegendMode, 441 setEgendMode, 441 setLegendMode, 441 rtti, 441 setLegendMode, 441 setPaintAttribute, 442 setPolygon, 442 setPolygon, 442 setPolygon, 442 setPolygon, 442 setPolygon, 442 setPolygon, 443 setShape, 443 setShape, 443 setShape, 443 testPaintAttribute, 443 colorMap, 445 drawDots, 446 drawSeries, 446 QwtPlotSpectroCurve, 445 drawDots, 446 QwtPlotSpectroCurve, 445 penWidth, 446 QwtPlotSpectroCurve, 445 rtti, 446 QwtPlotSpectroCurve, 445 rtti, 446 QwtPlotSpectroCurve, 445 rtti, 462 setPaintAttribute, 447 setColorMap, 447 setColorMap, 447 setPaintAttribute, 447 setSamples, 447, 448 testPaintAttribute, 448 CwtPlotSpectroCurve, 448 restPaintAttribute, 447 setSamples, 447, 448 testPaintAttribute, 448 CwtPlotSpectroGram, 448 colorMap, 450 contourLevels, 450 contourLevels, 450 contourPasterSize, 451 data, 451, 452 defaultContourPen, 452 lisplayMode, 456 lestContourLines, 452 interval, 453 setSymbolPen, 472 setSymbolExent, 472 setSymbolExent, 472 setSymbolExent, 472 setSymbolExent, 472 setSymbolExent, 472 setSymbolPen, 472 | | • | <u> </u> |
| draw, 440 legendlcon, 440 LegendMode, 439 legendMode, 449 legendMode, 440 PaintAttribute, 439 pen, 441 OwtPlotShapeltem, 439, 440 OwtPlotShapeltem, 439, 440 OwtPlotShapeltem, 439, 440 renderTolerance, 441 rti, 441 setBrush, 441 setBrush, 441 setPaintAttribute, 442 setPen, 442 setPen, 442 setPolygon, 442 setPolygon, 442 setRenderTolerance, 443 setShape, 443 testPaintAttribute, 443 OwtPlotSpectroCurve, 444 colorMap, 445 colorMap, 445 drawDots, 446 PaintAttribute, 445 penWidth, 446 OwtPlotSpectroCurve, 445 OwtPlotSpectroCurve, 446 OwtPlotSpectroCurve, 447 setPaintAttribute, 448 OwtPlotSpectroCurve, 448 OwtPlotSpectroCurve, 445 OwtPlotSpectroCurve, 445 OwtPlotSpectroCurve, 445 OwtPlotSpectroCurve, 445 OwtPlotSpectroCurve, 446 OwtPlotSpectroCurve, 445 OwtPlotSpectroCurve, 445 OwtPlotSpectroCurve, 446 OwtPlotSpectroCurve, 447 setPaintAttribute, 447 setPaintAttribute, 447 setPaintAttribute, 447 setPaintAttribute, 448 OwtPlotSpectroGyam, 448 OwtPlotTradingCurve, 466, 467 ortic, 466 OwtPlotTradingCurve, 466, 467 ortic, 469 owtPlotTradingCurve, 466, 467 ortic, 469 owtPlotTradingCurve, 466, 467 owtPlotTradingCurve, 466, 467 owtPlotTradingCurve, 466, 467 ortic, 469 owtPlotTradingCurve, 466, 467 owtPlotTradingCurve, | | · | |
| legendloon, 440 LegendlMode, 439 legendlMode, 440 PaintAttribute, 439 pen, 441 OwtPlotShapeltem, 439, 440 renderTolerance, 441 rtit, 441 setEagndlMode, 441 setEursh, 441 setEursh, 441 setEursh, 441 setPaintAttribute, 442 setPen, 442 setPolygon, 442 setPolygon, 442 setPolygon, 442 setRect, 443 setBrantAttribute, 443 setRect Tolerance, 443 setRect, 443 setRect Tolerance, 444 colorMap, 445 colorMap, 445 colorMap, 445 colorMap, 445 drawSeries, 446 PaintAttribute, 445 penWidth, 446 QwtPlotSpectroCurve, 445 QwtPlotTradingCurve, 466 QwtPlotSpectroCurve, 445 QwtPlotTradingCurve, 466 QwtPlotTradingCurve, 466, 467 QwtPlot | | | |
| LegendMode, 439 legendMode, 440 testDisplayMode, 456 PaintAttribute, 439 pen, 441 OwtPlotShapeltem, 439, 440 CowtPlotShapeltem, 439, 440 renderTolerance, 441 rtit, 441 setBrush, 441 setBrush, 441 setPaintAttribute, 442 setPen, 442 setPen, 442 setPen, 442 setPen, 442 setPen, 443 setBrush, 443 setBrush, 441 setBrush, 441 setBraintAttribute, 442 setPen, 442 setPen, 442 setPen, 443 setRenderTolerance, 443 setRenderTolerance, 443 setShape, 443 setShape, 443 setShape, 443 setShape, 443 setShape, 443 setShape, 445 colorRange, 445 drawSeries, 446 PaintAttribute, 445 OwtPlotSpectroCurve, 445 OwtPlotSpectroCurve, 445 OwtPlotSpectroCurve, 445 OwtPlotSpectroCurve, 445 OwtPlotSpectroCurve, 445 setColorRange, 447 setColorRange, 447 setColorRange, 447 setPaintAttribute, 447 setSamples, 447, 448 legendMode, 450 draw, 461 renderer, 459 rowtPlotTextLabel, 460 draw, 461 margin, 462 owtPlotTextLabel, 461 margin, 462 setMargin, 462 setText, 462 text, 462 drawSeries, 466 drawBar, 467 Direction, 466 drawBar, 467 drawCandleStick, 467 drawCandleStick, 467 drawSeries, 467 drawCandleStick, 467 drawSymboly, 468 legendloon, 468 rinkAttribute, 447 setSamples, 447, 448 renderer, 459 rendere | | | |
| legendMode, 440 Pain/Attribute, 439 pen, 441 QwtPlotShapeltem, 439, 440 renderTolerance, 441 rtt, 441 setBrush, 441 setBrush, 441 setPain/Attribute, 442 setPolygon, 442 setPolygon, 442 setPolygon, 442 setPect, 443 setBenderTolerance, 443 setShape, 443 setShape, 443 testPain/Attribute, 443 QwtPlotSpectroCurve, 444 colorMap, 445 colorMap, 445 drawSeries, 446 QwtPlotSpectroCurve, 445 qwtPlotTradingCurve, 466, 467 | | • | |
| PaintAttribute, 439 pen, 441 QwtPlotShapeltem, 439, 440 QwtPlotShapeltem, 439, 440 QwtPlotShapeltem, 439, 440 renderTolerance, 441 rtti, 441 setBrush, 441 setBrush, 441 setBrush, 441 setPaintAttribute, 442 setPolygon, 442 setPolygon, 442 setRect, 443 setRenderTolerance, 443 setRenderTolerance, 443 setRenderTolerance, 443 setRenderTolerance, 443 setRenderTolerance, 443 setRenderTolerance, 444 setShape, 443 colorMap, 445 colorMap, 445 drawSeries, 446 QwtPlotTextLabel, 461 qrawSeries, 446 QwtPlotSpectroCurve, 445 rtti, 446 estColorMap, 447 setColorRange, 447 setColorMap, 447 setColorRange, 447 setPaintAttribute, 448 colorMap, 447 setColorMap, 447 setPaintAttribute, 448 colorMap, 447 setPaintAttribute, 447 setPaintAttribute, 448 colorMap, 450 contourPen, 451 contourPen, 451 contourPen, 451 contourPen, 450 draw, 452 drawContourLines, 452 interval, 453 setSymbolBrush, 471 setSymbolBrush, 472 setSymbolPrund, 472 setSymbolPren, 472 | | | |
| pen, 441 | | | |
| OwtPlotShapeltem, 439, 440 OwtPlotShapeltem, 439, 440 OwtPlotShapeltem, 439, 440 renderTolerance, 441 rtit, 441 setBrush, 441 setBrush, 441 setLegendMode, 441 setPaintAttribute, 442 setPolygon, 442 setPolygon, 442 setRenderTolerance, 443 setShape, 443 setShape, 443 shape, 443 colorRange, 445 colorRange, 445 drawDots, 446 drawSeries, 446 PaintAttribute, 445 OwtPlotSpectroCurve, 445 OwtPlotSpectroCurve, 445 OwtPlotSpectroCurve, 445 colorRange, 447 setColorRange, 447 setColorRange, 447 setColorRange, 447 setColorRange, 447 setColorRange, 447 setPaintAttribute, 447 setPaintAttribute, 447 setPaintAttribute, 447 setPaintAttribute, 447 setPaintAttribute, 448 OwtPlotSpectroCurve, 448 colorRange, 447 setPaintAttribute, 448 colorRange, 450 contourLevels, 450 contourLevels, 450 contourLevels, 450 contourLevels, 450 contourLevels, 450 contourLevels, 450 draw, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 setSymbolPen, 472 | | | |
| OwtPlotShapeltem, 439, 440 renderTolerance, 441 rtti, 441 setBrush, 441 setBrush, 441 setBrush, 441 setPaintAttribute, 442 setPen, 442 setPolygon, 442 setRect, 443 setRenderTolerance, 443 setRenderTolerance, 443 setRenderTolerance, 443 setRenderTolerance, 444 setPaintAttribute, 443 colorMap, 445 colorMap, 445 drawDots, 446 drawSeries, 446 OwtPlotTsextLabel, 460 draw, 461 margin, 462 setMargin, 462 setMargin, 462 setMargin, 462 setMargin, 462 setMargin, 462 setMargin, 462 drawDots, 446 drawSeries, 446 OwtPlotSpectroCurve, 445 penWidth, 446 OwtPlotSpectroCurve, 445 penWidth, 446 OwtPlotSpectroCurve, 445 setColorMap, 447 setColorMap, 447 setPaintAttribute, 447 setPaintAttribute, 448 colorMap, 447 setPaintAttribute, 448 colorMap, 447 setPaintAttribute, 448 colorMap, 450 contourLevels, 450 contourLevels, 450 contourLevels, 450 contourLevels, 450 contourLevels, 450 contourLevels, 450 draw, 452 defaultContourLines, 452 interval, 453 load File, 459 QwtPlotSygettem, 458 render, 459 render, 459 render, 459 render, 459 render, 459 renderer, 459 render, 459 render, 459 render, 459 renderer, 459 render, 45 | | • | |
| renderTolerance, 441 rtti, 441 setEgendMode, 441 setLegendMode, 441 setLegendMode, 442 setPaintAttribute, 442 setPolygon, 442 setPolygon, 442 setRect, 443 setShape, 443 setShape, 443 testPaintAttribute, 444 colorMap, 445 colorRange, 445 drawSeries, 446 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 colorRange, 447 setColorRange, 447 setColorRange, 447 setColorRange, 447 setPaintAttribute, 448 QwtPlotSpectroCurve, 445 drawSeries, 467 drawSeries, 467 drawSeries, 467 drawSeries, 467 setColorMap, 447 setPaintAttribute, 447 setPaintAttribute, 447 setPaintAttribute, 448 QwtPlotSpectrogram, 448 colorMap, 450 contourLevels, 450 contourLevels, 450 contourLevels, 450 contourPen, 451 contourRasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 interval, 453 QwtPlotSpectrot, 472 setSymbolPen, 472 setSymbolExtent, 472 | | • | • |
| rtti, 441 setBrush, 441 setBrush, 441 setPaintAttribute, 442 setPolygon, 442 setPolygon, 442 setRenderTolerance, 443 setShape, 443 shape, 443 colorMap, 445 colorMap, 446 drawSeries, 446 PaintAttribute, 445 penWidth, 446 OwtPlotSpectroCurve, 445 penWidth, 446 OwtPlotSpectroCurve, 445 setColorMap, 447 setColorMap, 447 setColorMap, 447 setPaintAttribute, 448 OwtPlotSpectroCurve, 445 colorMap, 446 CowtPlotSpectroCurve, 445 colorMap, 447 setColorMap, 447 setColorMap, 447 setPaintAttribute, 447 setPaintAttribute, 447 setPaintAttribute, 447 setPaintAttribute, 447 setPaintAttribute, 447 setPaintAttribute, 448 OwtPlotSpectroCurve, 448 CowtPlotSpectroCurve, 445 colorMap, 447 setColorMap, 447 setColorMap, 447 setPaintAttribute, 447 setPaintAttribute, 448 colorMap, 450 contourLevels, 450 contourLevels, 450 contourPen, 451 contourRen, 451 contourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 setSymbolPen, 472 setSymbolEvent, 472 | | • | • |
| setBrush, 441 render, 459 setLegendMode, 441 renderer, 459 setPaintAttribute, 442 rtti, 459 setPen, 442 viewBox, 459 setPolygon, 442 OwtPlotTextLabel, 460 setRenderTolerance, 443 margin, 462 setShape, 443 OwtPlotTextLabel, 461 setShape, 443 OwtPlotTextLabel, 461 setShape, 443 OwtPlotTextLabel, 461 setPaintAttribute, 443 rti, 462 colorMap, 445 setMargin, 462 colorMap, 445 setText, 462 drawDots, 446 drawBeries, 462 drawBar, 465 dextRect, 462 drawBar, 466 QwtPlotTradingCurve, 464 boundingRect, 467 Direction, 466 drawBar, 467 Direction, 466 drawBar, 467 drawBar, 467 drawBar, 467 drawSeries, 467 setColorMap, 447 drawGardleStick, 467 rtti, 446 drawSeries, 467 setPaintAttribute, 447 legendlcon, 468 setPaintAttribute, 448 PaintAttribute, 468 QwtPlotSpectrogram, 448 QwtPlotTradingCurve | | | G . |
| setLegendMode, 441 setPaintAttribute, 442 setPen, 442 setPolygon, 442 setRect, 443 setRedt, 443 setRenderTolerance, 443 setShape, 443 setShape, 443 testPaintAttribute, 443 QwtPlotTextLabel, 460 draw, 461 margin, 462 QwtPlotTextLabel, 461 Titl, 462 QwtPlotTextLabel, 461 Titl, 462 QwtPlotTextLabel, 461 Titl, 462 QwtPlotSpectroCurve, 444 colorMap, 445 colorRange, 445 drawDots, 446 drawSeries, 446 PaintAttribute, 445 penWidth, 446 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 Titl, 446 SetColorMap, 447 QwtPlotSpectroCurve, 445 Titl, 446 SetColorMap, 447 SetColorMap, 447 setColorRange, 447 setPaintAttribute, 447 setPaintAttribute, 447 setPaintAttribute, 448 QwtPlotSpectroGram, 448 CowtPlotSpectroGram, 448 CowtPlotTradingCurve, 466, 467 ContourLevels, 450 contourPen, 451 contourPen, 451 contourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 setSymbolBrush, 471 setSymbolExtent, 472 setSymbolExtent, 472 setSymbolPen, 472 | | rtti, 441 | - |
| setPaintAttribute, 442 setPen, 442 setPolygon, 442 setRect, 443 setRenderTolerance, 443 setShape, 443 shape, 443 testPaintAttribute, 444 colorMap, 445 colorRange, 445 drawSeries, 446 QwtPlotSpectroCurve, 445 QwtPlotTextLabel, 461 rtti, 462 colorRange, 445 drawDots, 446 drawSeries, 446 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 QwtPlotTradingCurve, 464 DoundingRect, 467 Direction, 466 QwtPlotSpectroCurve, 445 GwtPlotSpectroCurve, 445 colorRange, 447 setColorRange, 447 setColorRange, 447 setPaintAttribute, 447 setPenWidth, 447 setPenWidth, 447 setPenWidth, 447 setPenWidth, 447 setPaintAttribute, 447 setPenWidth, 447 setPaintAttribute, 447 setPaintAttribute, 447 setPaintAttribute, 448 drawUserSymbol, 468 drawUserSymbol, 468 drawUserSymbol, 468 drawUserSymbol, 468 drawUserSymbolWidth, 469 minSymbolWidth, 469 minSymbolWidth, 469 paintAttribute, 446 QwtPlotSpectrogram, 448 colorMap, 450 contourLevels, 450 contourPasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 setSymbolPen, 472 | | setBrush, 441 | render, 459 |
| setPen, 442 setPolygon, 442 setRect, 443 setRenderTolerance, 443 setRenderTolerance, 443 setShape, 443 setShape, 443 testPaintAttribute, 443 QwtPlotTextLabel, 461 margin, 462 QwtPlotTextLabel, 461 rtti, 462 QwtPlotTextLabel, 461 rtti, 462 QwtPlotTextLabel, 461 rtti, 462 colorMap, 445 colorMap, 445 drawDots, 446 drawSeries, 446 PaintAttribute, 445 QwtPlotTradingCurve, 464 PaintAttribute, 445 penWidth, 446 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 rtti, 446 QwtPlotSpectroCurve, 445 rtti, 446 setColorMap, 447 setColorMap, 447 setPenWidth, 448 destPaintAttribute, 448 QwtPlotSpectrogram, 448 ColorMap, 450 contourLevels, 450 contourLevels, 450 contourRasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 viewBox, 459 QwtPlotTextLabel, 460 raw, 452 retx, 462 rett, 462 cott, 462 retx, 462 retw, 462 retx, 462 retw, 462 retw, 463 retw, 459 retx, 462 retw, 462 retw, 462 retx, 462 retw, 462 retw, 463 retw, 465 retx, 462 retw, 462 retw, 462 retw, 463 retw, 463 retw, 459 retw, 464 retw, 459 retw, 464 retw, 459 retw, 466 retw, 459 retw, 466 retw, 459 retw, 460 retw, 459 retw, 460 retw, 459 retw, 460 retw, 459 retw, 460 retw, 459 retw, 462 retw, | | setLegendMode, 441 | renderer, 459 |
| setPolygon, 442 setRect, 443 setRenderTolerance, 443 setRenderTolerance, 443 setShape, 443 setShape, 443 testPaintAttribute, 443 QwtPlotTextLabel, 461 testPaintAttribute, 443 QwtPlotSpectroCurve, 444 colorMap, 445 colorMap, 445 colorRange, 445 drawDots, 446 drawSeries, 446 PaintAttribute, 445 penWidth, 446 QwtPlotSpectroCurve, 445 qwtPlotSpectroCurve, 445 qwtPlotSpectroCurve, 445 penWidth, 446 QwtPlotSpectroCurve, 445 rtti, 446 setColorMap, 447 setColorRange, 447 setPenWidth, 447 setPenWidth, 447 setPaintAttribute, 448 QwtPlotSpectrogram, 448 colorMap, 450 contourPen, 451 contourRasterSize, 451 data, 451, 452 defaultContourLines, 452 interval, 453 QwtPlotSpect, 461 margin, 462 draw, 461 margin, 462 setMargin, 462 se | | setPaintAttribute, 442 | rtti, 459 |
| setRect, 443 setRenderTolerance, 443 setShape, 443 shape, 443 testPaintAttribute, 443 QwtPlotTextLabel, 461 tti, 462 QwtPlotSpectroCurve, 444 colorMap, 445 colorRange, 445 drawSeries, 446 PaintAttribute, 445 QwtPlotSpectroCurve, 446 PaintAttribute, 445 penWidth, 446 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 qwtPlotSpectroCurve, 445 qwtPlotSpectroCurve, 445 setColorRange, 447 setColorRange, 447 setPaintAttribute, 447 setPaintAttribute, 447 setPaintAttribute, 447 setPaintAttribute, 448 QwtPlotSpectroGram, 448 QwtPlotTradingCurve, 466, 467 colorMap, 450 contourPen, 451 contourRasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 setSymbolPen, 472 | | setPen, 442 | viewBox, 459 |
| setRenderTolerance, 443 setShape, 443 setShape, 443 destPaintAttribute, 443 QwtPlotTextLabel, 461 colorMap, 445 colorMap, 445 colorRange, 445 drawDots, 446 drawSeries, 446 PaintAttribute, 445 QwtPlotSpectroCurve, 445 penWidth, 446 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 drawBar, 467 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 drawBar, 467 QwtPlotSpectroCurve, 445 setColorMap, 447 setColorMap, 447 setPaintAttribute, 447 setPaintAttribute, 447 setPaintAttribute, 448 drawUserSymbol, 468 testPaintAttribute, 448 QwtPlotSpectrogram, 448 colorMap, 450 contourLevels, 450 contourPen, 451 contourRasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 margin, 462 QwtPlotTextLabel, 461 rtti, 462 cwtPlotTextLabel, 461 cwtPlotTextLabel, 461 cwtPlotTextLabel, 462 cwtPlotTextLabel, 461 cwtPlotTextLabel, 462 cwtPlotText, 462 cwtPlotTradingCurve, 466, 467 rtti, 469 setMaxSymbolWidth, 469 setMaxSymbolWidth, 469 setMinSymbolWidth, 469 setMinSymbolWidth, 471 setSamples, 471 setSamples, 471 setSamples, 471 setSymbolExtent, 472 setSymbolExtent, 472 setSymbolPen, 472 | | setPolygon, 442 | QwtPlotTextLabel, 460 |
| setRenderTolerance, 443 setShape, 443 setShape, 443 destPaintAttribute, 443 QwtPlotTextLabel, 461 colorMap, 445 colorMap, 445 colorRange, 445 drawDots, 446 drawSeries, 446 PaintAttribute, 445 QwtPlotSpectroCurve, 445 penWidth, 446 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 drawBar, 467 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 drawBar, 467 QwtPlotSpectroCurve, 445 setColorMap, 447 setColorMap, 447 setPaintAttribute, 447 setPaintAttribute, 447 setPaintAttribute, 448 drawUserSymbol, 468 testPaintAttribute, 448 QwtPlotSpectrogram, 448 colorMap, 450 contourLevels, 450 contourPen, 451 contourRasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 margin, 462 QwtPlotTextLabel, 461 rtti, 462 cwtPlotTextLabel, 461 cwtPlotTextLabel, 461 cwtPlotTextLabel, 462 cwtPlotTextLabel, 461 cwtPlotTextLabel, 462 cwtPlotText, 462 cwtPlotTradingCurve, 466, 467 rtti, 469 setMaxSymbolWidth, 469 setMaxSymbolWidth, 469 setMinSymbolWidth, 469 setMinSymbolWidth, 471 setSamples, 471 setSamples, 471 setSamples, 471 setSymbolExtent, 472 setSymbolExtent, 472 setSymbolPen, 472 | | • • | draw, 461 |
| setShape, 443 shape, 443 testPaintAttribute, 443 QwtPlotSpectroCurve, 444 colorMap, 445 drawDots, 446 drawDots, 446 PaintAttribute, 445 penWidth, 446 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 drawSeries, 446 PaintAttribute, 445 penWidth, 446 QwtPlotSpectroCurve, 445 drawSeries, 446 QwtPlotSpectroCurve, 445 drawSeries, 467 QwtPlotSpectroCurve, 445 drawSeries, 467 drawSaries, 467 drawSaries, 467 drawSeries, 467 drawSeries, 467 setColorMap, 447 setColorMap, 447 setPaintAttribute, 447 setPenWidth, 447 setSamples, 447, 448 testPaintAttribute, 448 QwtPlotSpectrogram, 448 QwtPlotTradingCurve, 466, 467 colorMap, 450 contourLevels, 450 contourPen, 451 contourRasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 QwtPlotTractinele, 461 setSymbolExtent, 472 setSymbolExtent, 472 setSymbolExtent, 472 setSymbolExtent, 472 setSymbolExtent, 472 | | | |
| shape, 443 testPaintAttribute, 443 QwtPlotSpectroCurve, 444 colorMap, 445 colorRange, 445 drawDots, 446 drawSeries, 446 PaintAttribute, 445 penWidth, 446 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 penWidth, 446 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 drawSeries, 467 QwtPlotSpectroCurve, 445 petColorMap, 447 setColorMap, 447 setPaintAttribute, 447 setSamples, 447, 448 testPaintAttribute, 448 QwtPlotSpectrogram, 448 QwtPlotTradingCurve, 466, 467 contourLevels, 450 contourPen, 451 contourRasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 QwtPlotTracing, 472 setSymbolExtent, 472 setSymbolExtent, 472 setSymbolExtent, 472 setSymbolExtent, 472 setSymbolExtent, 472 | | | G . |
| testPaintAttribute, 443 QwtPlotSpectroCurve, 444 colorMap, 445 colorRange, 445 drawDots, 446 drawDots, 446 drawSeries, 446 PaintAttribute, 445 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 qutPlotSpectroCurve, 447 qutPlotSpectroSpectroCurve, 448 qutPlotSpectroSp | | • | |
| OwtPlotSpectroCurve, 444 colorMap, 445 colorRange, 445 drawDots, 446 drawSeries, 446 PaintAttribute, 445 penWidth, 446 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 qrawBar, 467 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 qrawBar, 467 qrawBar, 467 qrawBar, 467 drawSeries, 467 drawSeries, 467 setColorMap, 447 setPenWidth, 447 setPenWidth, 448 QwtPlotSpectroCurve, 448 QwtPlotSpectroCurve, 445 minSymbolWidth, 469 testPaintAttribute, 447 setSamples, 447, 448 testPaintAttribute, 448 QwtPlotSpectrogram, 448 QwtPlotSpectrogram, 448 ColorMap, 450 contourPen, 451 contourPen, 451 contourPasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 setSymbolPen, 472 setSymbolExtent, 472 setSymbolExtent, 472 setSymbolPen, 472 | | • | |
| colorMap, 445 colorRange, 445 drawDots, 446 drawSeries, 446 PaintAttribute, 445 penWidth, 446 QwtPlotSpectroCurve, 445 setColorMap, 447 setColorRange, 447 setPaintAttribute, 448 drawSeries, 467 setColorRange, 447 setPaintAttribute, 448 drawUserSymbol, 468 testPaintAttribute, 448 drawUserSymbolWidth, 469 setSamples, 447, 448 testPaintAttribute, 448 QwtPlotSpectrogram, 448 colorMap, 450 contourLevels, 450 contourLevels, 450 data, 451, 452 defaultContourPen, 452 drawContourLines, 452 interval, 453 setText, 462 text, 467 drawBar, 467 drawBar, 467 drawBar, 467 drawSymbols, 468 drawCandleStick, 467 drawSymbolWidth, 469 minSymbolWidth, 469 minSymbolWidth, 469 PaintAttribute, 466 QwtPlotTradingCurve, 466, 467 colorMap, 450 contourPen, 451 setMaxSymbolWidth, 469 setMaxSymbolWidth, 469 setMaxSymbolWidth, 471 setSamples, 471 setSymbolBrush, 471 setSymbolBrush, 471 setSymbolExtent, 472 interval, 453 | | | |
| colorRarge, 445 drawDots, 446 drawSeries, 446 PaintAttribute, 445 penWidth, 446 QwtPlotSpectroCurve, 445 setColorMap, 447 setPaintAttribute, 447 setSamples, 447, 448 testPaintAttribute, 448 QwtPlotSpectrogram, 448 QwtPlotSpectrogram, 448 ColorMap, 450 contourPen, 451 contourPen, 452 drawContourLines, 452 interval, 453 text, 462 textRect, 462 QwtPlotTradingCurve, 464 boundingRect, 467 Direction, 466 drawBar, 467 drawSeries, 467 drawBar, 467 drawSarnbols, 468 drawSeries, 467 drawSeries, 467 drawSymbols, 468 drawUserSymbol, 468 drawUserSymbol, 468 egendlcon, 468 maxSymbolWidth, 469 minSymbolWidth, 469 minSymbolWidth, 469 PaintAttribute, 466 QwtPlotTradingCurve, 466, 467 rti, 469 setMaxSymbolWidth, 469 setMaxSymbolWidth, 469 setMaxSymbolWidth, 471 setSamples, 471 setSymbolBrush, 471 setSymbolBrush, 471 setSymbolExtent, 472 setSymbolPen, 472 | | · | G . |
| drawDots, 446 drawSeries, 446 QwtPlotTradingCurve, 464 PaintAttribute, 445 penWidth, 446 QwtPlotSpectroCurve, 445 getVelotSpectroCurve, 445 setColorMap, 447 setColorRange, 447 setPaintAttribute, 448 getPaintAttribute, 448 QwtPlotSpectroCurve, 445 getPaintAttribute, 447 setSamples, 447, 448 testPaintAttribute, 448 QwtPlotSpectrogram, 448 QwtPlotSpectrogram, 448 ColorMap, 450 contourLevels, 450 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 drawSeries, 467 drawSymbol, 468 drawUserSymbol, 468 drawUserSymbol, 468 drawUserSymbol, 468 elgendlcon, 468 maxSymbolWidth, 469 minSymbolWidth, 469 paintAttribute, 448 QwtPlotSpectrogram, 448 QwtPlotTradingCurve, 466, 467 contourLevels, 450 contourPen, 451 setMaxSymbolWidth, 469 setMaxSymbolWidth, 469 setMaxSymbolWidth, 469 setMaxSymbolWidth, 471 setSamples, 471 setSymbolBrush, 471 defaultContourLines, 452 interval, 453 setSymbolPen, 472 | | • | |
| drawSeries, 446 PaintAttribute, 445 penWidth, 446 OwtPlotSpectroCurve, 445 rtti, 446 SetColorMap, 447 setColorRange, 447 setPenWidth, 448 CowtPlotSpectroCurve, 445 setPenWidth, 447 setSamples, 447, 448 testPaintAttribute, 448 CwtPlotSpectrogram, 448 CwtPlotSpectrogram, 448 ColorMap, 450 contourPen, 451 contourRasterSize, 451 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLevels, 450 draw, 452 drawContourLevels, 450 draw, 452 drawContourLevels, 450 draw, 452 drawContourLevels, 450 draw, 452 interval, 453 OwtPlotTradingCurve, 464 boundingRect, 467 drawBoundingRect, 467 drawBoundingRect, 467 drawBoundingRect, 467 drawBoundingRect, 467 drawBoundingRect, 467 drawBoundingRect, 467 drawBar, 467 drawBoundingRect, 467 drawBoundingRect, 467 drawBoundingRect, 467 drawBar, 467 drawBar, 467 drawBar, 467 drawBoundingRect, 467 drawBar, 46 | | | |
| PaintAttribute, 445 penWidth, 446 | | • | |
| penWidth, 446 QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 rtti, 446 setColorMap, 447 setColorMange, 447 setPaintAttribute, 447 setPaintAttribute, 448 setPaintAttribute, 448 colorMap, 450 contourPen, 451 contourPen, 452 DisplayMode, 450 drawSeries, 467 drawSeries, 467 drawSymbols, 468 drawUserSymbol, 468 setPaintAttribute, 448 drawUserSymbol, 468 setPaintAttribute, 448 minSymbolWidth, 469 setSamples, 447, 448 paintAttribute, 466 QwtPlotTradingCurve, 466, 467 contourLevels, 450 contourPen, 451 contourPen, 451 setMaxSymbolWidth, 469 setMaxSymbolWidth, 469 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 setSymbolPen, 472 setSymbolPen, 472 | | | |
| QwtPlotSpectroCurve, 445 QwtPlotSpectroCurve, 445 rtti, 446 setColorMap, 447 setColorRange, 447 setPaintAttribute, 447 setSamples, 447, 448 testPaintAttribute, 448 QwtPlotSpectrogram, 448 ColorMap, 450 contourPen, 451 contourRasterSize, 451 defaultContourPen, 452 DisplayMode, 450 drawBar, 467 drawCandleStick, 467 drawSymbols, 468 drawUserSymbol, 468 drawUserSymbol, 468 legendlcon, 468 maxSymbolWidth, 469 minSymbolWidth, 469 minSymbolWidth, 469 paintAttribute, 466 QwtPlotTradingCurve, 466, 467 rtti, 469 scaledSymbolWidth, 469 setMaxSymbolWidth, 469 setMinSymbolWidth, 469 setSamples, 471 draw, 452 drawContourLines, 452 interval, 453 setSymbolPen, 472 | | • | _ |
| QwtPlotSpectroCurve, 445 rtti, 446 setColorMap, 447 setColorRange, 447 setPaintAttribute, 447 setSamples, 447, 448 testPaintAttribute, 448 QwtPlotSpectrogram, 448 colorMap, 450 contourLevels, 450 contourPen, 451 contourRasterSize, 451 defaultContourPen, 452 DisplayMode, 450 drawCandleStick, 467 drawSymbols, 468 drawSymbols, 468 drawUserSymbol, 468 legendlcon, 468 maxSymbolWidth, 469 minSymbolWidth, 469 minSymbolWidth, 469 each 466 QwtPlotTradingCurve, 466, 467 colorMap, 450 contourLevels, 450 contourPen, 451 setMaxSymbolWidth, 469 setMaxSymbolWidth, 469 setMinSymbolWidth, 469 setMaxSymbolWidth, 471 setSamples, 471 setSymbolBrush, 471 drawContourLines, 452 interval, 453 setSymbolPen, 472 | | • | |
| rtti, 446 setColorMap, 447 setColorRange, 447 setColorRange, 447 setPaintAttribute, 447 setPaintAttribute, 447 setSamples, 447, 448 testPaintAttribute, 448 QwtPlotSpectrogram, 448 colorMap, 450 contourLevels, 450 contourPen, 451 contourRasterSize, 451 defaultContourPen, 452 DisplayMode, 450 drawSeries, 467 drawSymbols, 468 drawUserSymbol, 468 drawUserSymbol, 468 drawUserSymbol, 468 drawUserSymbolWidth, 469 maxSymbolWidth, 469 minSymbolWidth, 469 QwtPlotTradingCurve, 466, 467 contourLevels, 450 rtti, 469 scaledSymbolWidth, 469 setMaxSymbolWidth, 469 setMaxSymbolWidth, 469 setMaxSymbolWidth, 471 setPaintAttribute, 471 setSamples, 471 draw, 452 drawContourLines, 452 interval, 453 setSymbolExtent, 472 interval, 453 | | | |
| setColorMap, 447 setColorRange, 447 setPaintAttribute, 447 setPenWidth, 447 setSamples, 447, 448 testPaintAttribute, 448 QwtPlotSpectrogram, 448 colorMap, 450 contourLevels, 450 contourRasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw Graw Symbols, 468 draw UserSymbol, 468 legendlcon, 468 maxSymbolWidth, 469 minSymbolWidth, 469 paintAttribute, 466 QwtPlotTradingCurve, 466, 467 cwtPlotTradingCurve, 466, 467 rtti, 469 scaledSymbolWidth, 469 setMaxSymbolWidth, 469 setMaxSymbolWidth, 469 setMaxSymbolWidth, 471 setSamples, 471 setSymbolBrush, 471 setSymbolBrush, 471 setSymbolExtent, 472 interval, 453 setSymbolPen, 472 | | • | |
| setColorRange, 447 setPaintAttribute, 447 setPenWidth, 447 setSamples, 447, 448 testPaintAttribute, 448 QwtPlotSpectrogram, 448 QwtPlotTradingCurve, 466, 467 colorMap, 450 contourLevels, 450 contourPen, 451 contourRasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 drawUserSymbol, 468 legendlcon, 468 maxSymbolWidth, 469 minSymbolWidth, 469 PaintAttribute, 466 QwtPlotTradingCurve, 466, 467 cwtPlotTradingCurve, 466, 467 rtti, 469 scaledSymbolWidth, 469 setMaxSymbolWidth, 469 setMaxSymbolWidth, 471 setSamples, 471 setSamples, 471 setSymbolBrush, 471 setSymbolExtent, 472 setSymbolPen, 472 | | | |
| setPaintAttribute, 447 setPenWidth, 447 setSamples, 447, 448 testPaintAttribute, 448 QwtPlotSpectrogram, 448 ColorMap, 450 contourLevels, 450 contourPen, 451 contourRasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 legendlcon, 468 maxSymbolWidth, 469 minSymbolWidth, 469 PaintAttribute, 466 QwtPlotTradingCurve, 466, 467 QwtPlotTradingCurve, 466, 467 rtti, 469 scaledSymbolWidth, 469 setMaxSymbolWidth, 469 setMaxSymbolWidth, 471 setSamples, 471 setSamples, 471 setSymbolBrush, 471 setSymbolExtent, 472 setSymbolPen, 472 | | • • | • |
| setPenWidth, 447 setSamples, 447, 448 testPaintAttribute, 448 QwtPlotSpectrogram, 448 colorMap, 450 contourLevels, 450 contourPen, 451 contourRasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 interval, 453 maxSymbolWidth, 469 minSymbolWidth, 469 minSymbolWidth, 469 caledSymbolTradingCurve, 466, 467 rtti, 469 cwtPlotTradingCurve, 466, 467 rtti, 469 scaledSymbolWidth, 469 setMaxSymbolWidth, 469 setMinSymbolWidth, 471 setSamples, 471 setSymbolBrush, 471 setSymbolBrush, 471 setSymbolExtent, 472 setSymbolPen, 472 | | 3 , | • |
| setSamples, 447, 448 testPaintAttribute, 448 QwtPlotSpectrogram, 448 ColorMap, 450 contourLevels, 450 contourPen, 451 contourRasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 interval, 453 minSymbolWidth, 469 PaintAttribute, 466 QwtPlotTradingCurve, 466, 467 rtti, 469 scaledSymbolWidth, 469 scaledSymbolWidth, 469 setMaxSymbolWidth, 469 setMinSymbolWidth, 471 setSamples, 471 setSymbolBrush, 471 setSymbolBrush, 471 setSymbolExtent, 472 setSymbolPen, 472 | | setPaintAttribute, 447 | legendlcon, 468 |
| testPaintAttribute, 448 QwtPlotSpectrogram, 448 QwtPlotTradingCurve, 466, 467 colorMap, 450 contourLevels, 450 contourPen, 451 contourRasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 interval, 453 PaintAttribute, 466 QwtPlotTradingCurve, 466, 467 rtti, 469 scaledSymbolWidth, 469 scaledSymbolWidth, 469 setMaxSymbolWidth, 469 setMinSymbolWidth, 471 setPaintAttribute, 471 setSymbolBrush, 471 setSymbolBrush, 471 setSymbolExtent, 472 setSymbolPen, 472 | | setPenWidth, 447 | maxSymbolWidth, 469 |
| QwtPlotSpectrogram, 448 | | setSamples, 447, 448 | minSymbolWidth, 469 |
| colorMap, 450 contourLevels, 450 contourPen, 451 contourRasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 QwtPlotTradingCurve, 466, 467 rtti, 469 scaledSymbolWidth, 469 setMaxSymbolWidth, 469 setMinSymbolWidth, 471 setSamples, 471 setSymbolBrush, 471 setSymbolBrush, 471 setSymbolExtent, 472 setSymbolPen, 472 | | testPaintAttribute, 448 | PaintAttribute, 466 |
| colorMap, 450 contourLevels, 450 contourPen, 451 contourRasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 QwtPlotTradingCurve, 466, 467 rtti, 469 scaledSymbolWidth, 469 setMaxSymbolWidth, 469 setMinSymbolWidth, 471 setSamples, 471 setSymbolBrush, 471 setSymbolBrush, 471 setSymbolExtent, 472 setSymbolPen, 472 | QwtF | PlotSpectrogram, 448 | QwtPlotTradingCurve, 466, 467 |
| contourLevels, 450 contourPen, 451 contourRasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 rtti, 469 scaledSymbolWidth, 469 setMaxSymbolWidth, 469 setMinSymbolWidth, 471 setPaintAttribute, 471 setSymbolBrush, 471 setSymbolBrush, 471 setSymbolExtent, 472 setSymbolPen, 472 | | | QwtPlotTradingCurve, 466, 467 |
| contourPen, 451 contourRasterSize, 451 data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 scaledSymbolWidth, 469 setMaxSymbolWidth, 469 setMinSymbolWidth, 471 setPaintAttribute, 471 setSymbolBrush, 471 setSymbolExtent, 472 setSymbolPen, 472 | | • | _ |
| contourRasterSize, 451 data, 451, 452 setMaxSymbolWidth, 469 setMinSymbolWidth, 471 setPaintAttribute, 471 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 setMaxSymbolWidth, 469 setMinSymbolWidth, 471 setPaintAttribute, 471 setSamples, 471 setSymbolBrush, 471 setSymbolExtent, 472 setSymbolPen, 472 | | | scaledSymbolWidth, 469 |
| data, 451, 452 defaultContourPen, 452 DisplayMode, 450 draw, 452 drawContourLines, 452 interval, 453 setMinSymbolWidth, 471 setPaintAttribute, 471 setSamples, 471 setSymbolBrush, 471 setSymbolExtent, 472 setSymbolPen, 472 | | | |
| defaultContourPen, 452 setPaintAttribute, 471 DisplayMode, 450 setSamples, 471 draw, 452 setSymbolBrush, 471 drawContourLines, 452 setSymbolExtent, 472 interval, 453 setSymbolPen, 472 | | | • |
| DisplayMode, 450 setSamples, 471 draw, 452 setSymbolBrush, 471 drawContourLines, 452 setSymbolExtent, 472 interval, 453 setSymbolPen, 472 | | | - |
| draw, 452 setSymbolBrush, 471 drawContourLines, 452 setSymbolExtent, 472 interval, 453 setSymbolPen, 472 | | | |
| drawContourLines, 452 setSymbolExtent, 472 interval, 453 setSymbolPen, 472 | | • • | • |
| interval, 453 setSymbolPen, 472 | | | |
| · | | | |
| pixeimin, 453 setsymbolstyle, 472 | | | • |
| | | pixeiniii, 400 | setoyinbolotyle, 4/2 |

| symbolBrush, 473 | QwtPoint3DSeriesData, 489 |
|------------------------------|---------------------------|
| symbolExtent, 473 | QwtPointArrayData, 489 |
| symbolPen, 473 | boundingRect, 490 |
| SymbolStyle, 466 | QwtPointArrayData, 490 |
| symbolStyle, 473 | QwtPointArrayData, 490 |
| testPaintAttribute, 473 | sample, 490 |
| QwtPlotZoneItem, 474 | size, 491 |
| boundingRect, 475 | xData, 491 |
| brush, 475 | yData, 491 |
| draw, 475 | QwtPointMapper, 491 |
| interval, 475 | boundingRect, 492 |
| orientation, 476 | flags, 492 |
| pen, 476 | setBoundingRect, 493 |
| • | _ |
| QwtPlotZoneItem, 475 | setFlag, 493 |
| QwtPlotZoneItem, 475 | setFlags, 493 |
| rtti, 476 | testFlag, 493 |
| setBrush, 476 | tolmage, 494 |
| setInterval, 476 | toPoints, 494 |
| setOrientation, 477 | toPointsF, 494 |
| setPen, 477 | toPolygon, 495 |
| QwtPlotZoomer, 477 | toPolygonF, 495 |
| accept, 480 | TransformationFlag, 492 |
| begin, 481 | TransformationFlags, 492 |
| end, 481 | QwtPointPolar, 496 |
| maxStackDepth, 481 | normalized, 497 |
| minZoomSize, 481 | operator==, 498 |
| moveBy, 481 | QwtPointPolar, 497 |
| moveTo, 482 | QwtPointPolar, 497 |
| QwtPlotZoomer, 480 | setPoint, 498 |
| QwtPlotZoomer, 480 | toPoint, 498 |
| rescale, 482 | QwtPointSeriesData, 499 |
| setAxis, 482 | boundingRect, 500 |
| setMaxStackDepth, 482 | QwtPointSeriesData, 499 |
| setZoomBase, 482, 484 | QwtPointSeriesData, 499 |
| setZoomStack, 484 | QwtPowerTransform, 500 |
| widgetKeyPressEvent, 484 | copy, 502 |
| | invTransform, 502 |
| widgetMouseReleaseEvent, 484 | QwtPowerTransform, 501 |
| zoom, 485 | QwtPowerTransform, 501 |
| zoomBase, 485 | , , |
| zoomRect, 485 | transform, 502 |
| zoomRectIndex, 485 | QwtRasterData, 502 |
| zoomStack, 486 | ConrecFlag, 503 |
| zoomed, 485 | contourLines, 503 |
| QwtPoint3D, 486 | discardRaster, 504 |
| isNull, 487 | initRaster, 504 |
| operator==, 487 | interval, 504 |
| QwtPoint3D, 487 | pixelHint, 504 |
| QwtPoint3D, 487 | setInterval, 505 |
| rx, 487 | value, <u>505</u> |
| ry, 487 | QwtRichTextEngine, 505 |
| rz, 487 | draw, 506 |
| toPoint, 487 | heightForWidth, 506 |
| x, 488 | mightRender, 507 |
| y, 488 | textMargins, 507 |
| z, 488 | textSize, 507 |
| QwtPoint3DSeriesData, 488 | QwtRoundScaleDraw, 507 |
| boundingRect, 489 | drawBackbone, 509 |
| QwtPoint3DSeriesData, 489 | drawLabel, 509 |
| 2 22 22.102 attag 100 | |

| drawTick, 509 | move, 526, 528 |
|--------------------------|---------------------------------------|
| extent, 509 | orientation, 528 |
| moveCenter, 510 | pos, 528 |
| QwtRoundScaleDraw, 509 | QwtScaleDraw, 522 |
| QwtRoundScaleDraw, 509 | QwtScaleDraw, 522 |
| radius, 510 | setAlignment, 528 |
| setAngleRange, 510 | setLabelAlignment, 529 |
| setRadius, 510 | setLabelRotation, 529 |
| QwtSamplingThread, 511 | setLength, 529 |
| elapsed, 512 | QwtScaleEngine, 530 |
| interval, 512 | Attribute, 531 |
| run, 512 | attributes, 532 |
| sample, 512 | autoScale, 532 |
| setInterval, 512 | base, 532 |
| stop, 513 | buildInterval, 532 |
| QwtScaleArithmetic, 513 | contains, 532 |
| ceilEps, 513 | divideInterval, 533 |
| divideEps, 513 | divideScale, 533 |
| divideInterval, 514 | lowerMargin, 533 |
| floorEps, 514 | QwtScaleEngine, 531 |
| QwtScaleDiv, 514 | QwtScaleEngine, 531 |
| bounded, 517 | reference, 533 |
| contains, 518 | setAttribute, 533 |
| interval, 518 | setAttributes, 534 |
| invert, 518 | setBase, 534 |
| inverted, 518 | setMargins, 534 |
| lowerBound, 518 | setReference, 534 |
| operator==, 519 | setTransformation, 535 |
| QwtScaleDiv, 516, 517 | strip, 535 |
| QwtScaleDiv, 516, 517 | testAttribute, 535 |
| range, 519 | transformation, 535 |
| setInterval, 519 | upperMargin, 535 |
| setLowerBound, 519 | QwtScaleMap, 536 |
| setTicks, 519 | ~QwtScaleMap, 537 |
| setUpperBound, 520 | invTransform, 537 |
| TickType, 515 | isInverting, 538 |
| ticks, 520 | p1, 538 |
| upperBound, 520 | p2, 538 |
| QwtScaleDraw, 520 | pDist, 538 |
| Alignment, 522 | QwtScaleMap, 537 |
| alignment, 522 | QwtScaleMap, 537 |
| boundingLabelRect, 523 | s1, 538 |
| drawBackbone, 523 | s2, 538 |
| drawLabel, 523 | sDist, 538 |
| drawTick, 523 | setPaintInterval, 538 |
| extent, 524 | setScaleInterval, 539 |
| | · · · · · · · · · · · · · · · · · · · |
| getBorderDistHint, 524 | setTransformation, 539 |
| labelAlignment, 524 | transform, 539 |
| labelPosition, 524 | QwtScaleWidget, 540 |
| labelRect, 525 | alignment, 542 |
| labelRotation, 525 | colorBarInterval, 543 |
| labelSize, 525 | colorBarRect, 543 |
| labelTransformation, 525 | colorBarWidth, 543 |
| length, 525 | colorMap, 543 |
| maxLabelHeight, 526 | dimForLength, 543 |
| maxLabelWidth, 526 | drawColorBar, 543 |
| minLabelDist, 526 | drawTitle, 544 |
| minLength, 526 | endBorderDist, 544 |

| getBorderDistHint, 544 | drawRose, 560 |
|---------------------------|---------------------------|
| getMinBorderDist, 544 | numThornLevels, 560 |
| isColorBarEnabled, 545 | numThorns, 561 |
| LayoutFlag, 542 | QwtSimpleCompassRose, 560 |
| layoutScale, 545 | QwtSimpleCompassRose, 560 |
| margin, 545 | setNumThornLevels, 561 |
| minimumSizeHint, 545 | setNumThorns, 561 |
| QwtScaleWidget, 542 | setShrinkFactor, 561 |
| QwtScaleWidget, 542 | setWidth, 561 |
| resizeEvent, 545 | shrinkFactor, 562 |
| scaleChange, 545 | width, 562 |
| scaleDraw, 545, 546 | QwtSlider, 562 |
| setAlignment, 546 | borderWidth, 564 |
| setBorderDist, 546 | changeEvent, 565 |
| setColorBarEnabled, 546 | drawHandle, 565 |
| setColorBarWidth, 546 | drawSlider, 565 |
| setColorMap, 547 | handleRect, 565 |
| setLabelAlignment, 547 | handleSize, 565 |
| setLabelRotation, 547 | hasGroove, 565 |
| setLayoutFlag, 547 | hasTrough, 565 |
| setMargin, 547 | isScrollPosition, 566 |
| setMinBorderDist, 548 | minimumSizeHint, 566 |
| setScaleDiv, 548 | mousePressEvent, 566 |
| | mouseReleaseEvent, 566 |
| setScaleDraw, 548 | |
| setSpacing, 548 | orientation, 566 |
| setTitle, 549 | paintEvent, 567 |
| setTransformation, 549 | QwtSlider, 564 |
| sizeHint, 549 | QwtSlider, 564 |
| spacing, 549 | resizeEvent, 567 |
| startBorderDist, 549 | scaleDraw, 567 |
| testLayoutFlag, 550 | ScalePosition, 564 |
| title, 550 | scalePosition, 567 |
| titleHeightForWidth, 550 | scrolledTo, 567 |
| QwtSeriesData | setBorderWidth, 567 |
| boundingRect, 552 | setGroove, 568 |
| sample, 552 | setHandleSize, 568 |
| setRectOfInterest, 552 | setOrientation, 568 |
| size, 552 | setScaleDraw, 568 |
| QwtSeriesData< T >, 550 | setScalePosition, 569 |
| QwtSeriesStore | setSpacing, 569 |
| data, 553 | setTrough, 569 |
| dataRect, 554 | setUpdateInterval, 569 |
| dataSize, 554 | sizeHint, 570 |
| sample, 554 | sliderRect, 570 |
| setData, 554 | spacing, 570 |
| setRectOfInterest, 554 | timerEvent, 570 |
| swapData, 555 | updateInterval, 570 |
| QwtSeriesStore< T >, 552 | QwtSpline, 570 |
| QwtSetSample, 555 | buildNaturalSpline, 572 |
| added, 557 | buildPeriodicSpline, 572 |
| QwtSetSample, 555 | coefficientsA, 572 |
| QwtSetSample, 555 | coefficientsB, 572 |
| QwtSetSeriesData, 557 | coefficientsC, 572 |
| boundingRect, 559 | operator=, 572 |
| QwtSetSeriesData, 557 | points, 573 |
| QwtSetSeriesData, 557 | QwtSpline, 572 |
| QwtSimpleCompassRose, 559 | QwtSpline, 572 |
| draw, 560 | setPoints, 573 |
| , | 23 |

| setSplineType, 573 | QwtSystemClock, 591 |
|----------------------------|--------------------------|
| SplineType, 572 | elapsed, 592 |
| splineType, 573 | isNull, 592 |
| value, 573 | restart, 592 |
| QwtSplineCurveFitter, 574 | start, 592 |
| fitCurve, 575 | QwtText, 592 |
| FitMode, 575 | backgroundBrush, 595 |
| fitMode, 575 | borderPen, 595 |
| setFitMode, 575 | borderRadius, 595 |
| setSpline, 575 | draw, 596 |
| setSplineSize, 576 | heightForWidth, 596 |
| spline, 576 | isEmpty, 596 |
| splineSize, 576 | isNull, 596 |
| • | |
| QwtSymbol, 576 | LayoutAttribute, 594 |
| boundingRect, 580 | PaintAttribute, 594 |
| brush, 580 | QwtText, 595 |
| CachePolicy, 578 | QwtText, 595 |
| cachePolicy, 580 | renderFlags, 596 |
| drawSymbol, 581 | setBackgroundBrush, 596 |
| drawSymbols, 581 | setBorderPen, 597 |
| graphic, 581 | setBorderRadius, 597 |
| invalidateCache, 582 | setColor, 597 |
| isPinPointEnabled, 582 | setFont, 597 |
| path, 582 | setLayoutAttribute, 597 |
| pen, 582 | setPaintAttribute, 598 |
| pinPoint, 582 | setRenderFlags, 598 |
| pixmap, 582 | setText, 598 |
| QwtSymbol, 579, 580 | setTextEngine, 598 |
| QwtSymbol, 579, 580 | testLayoutAttribute, 599 |
| renderSymbols, 583 | testPaintAttribute, 599 |
| setBrush, 583 | text, 599 |
| setCachePolicy, 583 | textEngine, 599, 600 |
| setColor, 583 | TextFormat, 594 |
| | |
| setGraphic, 584 | textSize, 600 |
| setPath, 584 | usedColor, 600 |
| setPen, 585 | usedFont, 600 |
| setPinPoint, 585 | QwtTextEngine, 601 |
| setPinPointEnabled, 585 | draw, 602 |
| setPixmap, 586 | heightForWidth, 602 |
| setSize, 586 | mightRender, 602 |
| setStyle, 586 | textMargins, 602 |
| setSvgDocument, 587 | textSize, 603 |
| size, 587 | QwtTextLabel, 603 |
| Style, 578 | heightForWidth, 605 |
| style, 587 | paintEvent, 605 |
| QwtSyntheticPointData, 587 | plainText, 605 |
| boundingRect, 589 | QwtTextLabel, 604 |
| interval, 589 | QwtTextLabel, 604 |
| QwtSyntheticPointData, 589 | setIndent, 605 |
| QwtSyntheticPointData, 589 | setMargin, 605 |
| rectOfInterest, 589 | setPlainText, 605 |
| sample, 590 | setText, 605, 607 |
| setInterval, 590 | textRect, 607 |
| setRectOfInterest, 590 | QwtThermo, 607 |
| setSize, 590 | alarmBrush, 610 |
| size, 591 | alarmEnabled, 610 |
| | alarmLevel, 610 |
| x, 591 | |
| y, 591 | alarmRect, 611 |

| borderWidth, 611 | isInverted, 627 |
|----------------------------|--------------------------|
| changeEvent, 611 | isTracking, 627 |
| colorMap, 611 | keyPressEvent, 627 |
| drawLiquid, 611 | mass, 628 |
| fillBrush, 613 | maximum, 628 |
| fillRect, 613 | minimum, 628 |
| minimumSizeHint, 613 | minimumSizeHint, 628 |
| orientation, 613 | mouseMoveEvent, 628 |
| origin, 613 | mousePressEvent, 629 |
| OriginMode, 609 | mouseReleaseEvent, 629 |
| originMode, 614 | orientation, 629 |
| paintEvent, 614 | pageStepCount, 629 |
| pipeRect, 614 | paintEvent, 629 |
| pipeWidth, 614 | setBorderWidth, 629 |
| QwtThermo, 610 | setInverted, 630 |
| QwtThermo, 610 | setMass, 630 |
| rangeFlags, 614 | setMaximum, 630 |
| resizeEvent, 614 | setMinimum, 630 |
| scaleDraw, 615 | setOrientation, 631 |
| ScalePosition, 609 | setPageStepCount, 631 |
| scalePosition, 615 | setRange, 631 |
| setAlarmBrush, 615 | setSingleStep, 631 |
| setAlarmEnabled, 615 | setStrepAlignment, 632 |
| | |
| setAlarmLevel, 616 | setTickCount, 632 |
| setBorderWidth, 616 | setTotalAngle, 632 |
| setColorMap, 616 | setTracking, 632 |
| setFillBrush, 616 | setUpdateInterval, 633 |
| setOrientation, 617 | setValue, 633 |
| setOrigin, 617 | setViewAngle, 633 |
| setOriginMode, 617 | setWheelBorderWidth, 633 |
| setPipeWidth, 617 | setWheelWidth, 634 |
| setRangeFlags, 617 | setWrapping, 634 |
| setScaleDraw, 618 | singleStep, 634 |
| setScalePosition, 618 | sizeHint, 634 |
| setSpacing, 618 | stepAlignment, 634 |
| setValue, 618 | tickCount, 634 |
| sizeHint, 618 | timerEvent, 635 |
| spacing, 619 | totalAngle, 635 |
| QwtTradingChartData, 619 | updateInterval, 635 |
| boundingRect, 620 | value, 635 |
| QwtTradingChartData, 620 | valueAt, 635 |
| QwtTradingChartData, 620 | valueChanged, 636 |
| QwtTransform, 620 | viewAngle, 636 |
| bounded, 621 | wheelBorderWidth, 636 |
| invTransform, 621 | wheelEvent, 636 |
| transform, 621 | wheelMoved, 636 |
| QwtWeedingCurveFitter, 622 | wheelPressed, 637 |
| chunkSize, 623 | wheelRect, 637 |
| fitCurve, 623 | wheelReleased, 637 |
| QwtWeedingCurveFitter, 623 | wheelWidth, 637 |
| QwtWeedingCurveFitter, 623 | wrapping, 637 |
| setChunkSize, 623 | QwtWidgetOverlay, 637 |
| setTolerance, 624 | drawOverlay, 640 |
| tolerance, 624 | eventFilter, 640 |
| QwtWheel, 624 | maskHint, 640 |
| borderWidth, 627 | MaskMode, 639 |
| drawTicks, 627 | maskMode, 640 |
| drawWheelBackground, 627 | paintEvent, 641 |
| ~ | , |

| QwtWidgetOverlay, 640 | QwtPlotSpectrogram, 453 |
|--------------------------------------|--------------------------|
| QwtWidgetOverlay, 640 | renderDocument |
| RenderMode, 639 | QwtPlotRenderer, 415 |
| renderMode, 641 | renderFlags |
| resizeEvent, 641 | QwtText, 596 |
| setMaskMode, 641 | renderFooter |
| setRenderMode, 641 | QwtPlotRenderer, 416 |
| updateOverlay, 642 | RenderHint |
| | QwtGraphic, 142 |
| RGB | QwtPlotItem, 352 |
| QwtColorMap, 74 | RenderHints |
| RTriangle | QwtGraphic, 142 |
| QwtSymbol, 579 | renderImage |
| radius | QwtPlotRasterItem, 410 |
| QwtRoundScaleDraw, 510 | QwtPlotSpectrogram, 454 |
| Raised | renderItem |
| QwtColumnSymbol, 77 | QwtLegend, 181 |
| QwtDial, 117 | renderLegend |
| QwtKnob, 169 | QwtAbstractLegend, 33 |
| QwtPlotGLCanvas, 323 | QwtLegend, 181 |
| range | QwtPlotRenderer, 416 |
| QwtScaleDiv, 519 | RenderMode |
| rangeFlags | QwtWidgetOverlay, 639 |
| QwtThermo, 614 | renderMode |
| Ray | QwtWidgetOverlay, 641 |
| QwtDialSimpleNeedle, 127 | renderScale |
| ReadOnly | QwtPlotRenderer, 416 |
| QwtLegendData, 183 | renderSymbols |
| Rect | QwtSymbol, 583 |
| QwtSymbol, 579 | renderThreadCount |
| rect | QwtPlotItem, 356 |
| QwtPixelMatrix, 261 | renderTile |
| RectRubberBand | QwtPlotSpectrogram, 454 |
| QwtPicker, 238 | renderTitle |
| RectSelection QwtPickerMachine, 257 | QwtPlotRenderer, 417 |
| rectOfInterest | renderTo |
| QwtSyntheticPointData, 589 | QwtPlotRenderer, 417 |
| | renderTolerance |
| reference QwtScaleEngine, 533 | QwtPlotShapeltem, 441 |
| referenceAxis | renderer |
| QwtPlotRescaler, 425 | QwtPlotSvgltem, 459 |
| remove | replot |
| QwtPicker, 242 | QwtPlot, 278 |
| removeItem | QwtPlotCanvas, 300 |
| QwtPlotDict, 318 | ResampleMode |
| removed | QwtMatrixRasterData, 212 |
| QwtPicker, 243 | resampleMode |
| render | QwtMatrixRasterData, 212 |
| QwtGraphic, 148 | rescale |
| QwtPlotRenderer, 415 | QwtAbstractScale, 38 |
| QwtPlotSvgItem, 459 | QwtMagnifier, 204 |
| RenderAntialiased | QwtPlotMagnifier, 383 |
| QwtPlotItem, 352 | QwtPlotRescaler, 425 |
| RenderPensUnscaled | QwtPlotZoomer, 482 |
| QwtGraphic, 142 | RescalePolicy |
| renderCanvas | QwtPlotRescaler, 421 |
| QwtPlotRenderer, 415 | rescalePolicy |
| renderContourLines | QwtPlotRescaler, 425 |
| | , |

| | Diri Di ib ol |
|---------------------------|------------------------|
| reset | Rtti_PlotBarChart |
| QwtGraphic, 148 | QwtPlotItem, 352 |
| QwtPicker, 243 | Rtti_PlotCurve |
| resetSymbolMap | QwtPlotItem, 352 |
| QwtPlotMultiBarChart, 395 | Rtti_PlotGrid |
| resizeEvent | QwtPlotItem, 352 |
| QwtPlot, 278 | Rtti_PlotHistogram |
| QwtPlotCanvas, 300 | QwtPlotItem, 352 |
| QwtScaleWidget, 545 | Rtti_PlotIntervalCurve |
| QwtSlider, 567 | QwtPlotItem, 352 |
| QwtThermo, 614 | Rtti_PlotItem |
| QwtWidgetOverlay, 641 | QwtPlotItem, 352 |
| ResizeMode | Rtti_PlotLegend |
| QwtPicker, 237 | QwtPlotItem, 352 |
| resizeMode | Rtti PlotMarker |
| QwtPicker, 243 | QwtPlotItem, 352 |
| restart | Rtti PlotMultiBarChart |
| QwtSystemClock, 592 | QwtPlotItem, 353 |
| • | Rtti PlotSVG |
| rgb | QwtPlotItem, 352 |
| QwtAlphaColorMap, 62 | Rtti PlotScale |
| QwtColorMap, 75 | QwtPlotItem, 352 |
| QwtLinearColorMap, 192 | Rtti_PlotShape |
| RichText | QwtPlotItem, 353 |
| QwtText, 595 | Rtti_PlotSpectroCurve |
| RightLegend | _ · |
| QwtPlot, 268 | QwtPlotItem, 352 |
| RightScale | Rtti_PlotSpectrogram |
| QwtScaleDraw, 522 | QwtPlotItem, 352 |
| RightToLeft | Rtti_PlotTextLabel |
| QwtColumnRect, 76 | QwtPlotItem, 353 |
| rose | Rtti_PlotTradingCurve |
| QwtCompass, 81, 82 | QwtPlotItem, 352 |
| RotateNeedle | Rtti_PlotUserItem |
| QwtDial, 117 | QwtPlotItem, 353 |
| RotateScale | Rtti_PlotZone |
| QwtDial, 117 | QwtPlotItem, 353 |
| RoundPoints | RttiValues |
| QwtPointMapper, 492 | QwtPlotItem, 352 |
| roundingAlignment | RubberBand |
| | QwtPicker, 238 |
| QwtPainter, 224 | rubberBand |
| rtti | QwtPicker, 243 |
| QwtPlotBarChart, 294 | rubberBandMask |
| QwtPlotCurve, 311 | QwtPicker, 243 |
| QwtPlotGrid, 329 | rubberBandOverlay |
| QwtPlotHistogram, 338 | QwtPicker, 243 |
| QwtPlotIntervalCurve, 345 | rubberBandPen |
| QwtPlotItem, 356 | QwtPicker, 243 |
| QwtPlotLegendItem, 378 | run |
| QwtPlotMarker, 387 | QwtSamplingThread, 512 |
| QwtPlotMultiBarChart, 395 | rx |
| QwtPlotScaleItem, 432 | QwtPoint3D, 487 |
| QwtPlotShapeItem, 441 | ry |
| QwtPlotSpectroCurve, 446 | QwtPoint3D, 487 |
| QwtPlotSpectrogram, 454 | |
| QwtPlotSvgItem, 459 | rz OwtPoint3D 487 |
| QwtPlotTextLabel, 462 | QwtPoint3D, 487 |
| QwtPlotTradingCurve, 469 | s1 |
| QwtPlotZoneItem, 476 | QwtScaleMap, 538 |
| aa | antodiomap, ooo |

| 2 | OutPlot ovout 269 |
|---|--|
| s2 QwtScaleMap, 538 | QwtPlotLayout, 368 QwtPlotPicker, 405 |
| sDist | scaleStepSize |
| QwtScaleMap, 538 | QwtAbstractScale, 39 |
| sample | ScaledColors |
| QwtArraySeriesData, 68 | QwtLinearColorMap, 191 |
| QwtCPointerData, 98 | scaledBoundingRect |
| QwtPointArrayData, 490 | QwtGraphic, 149 |
| QwtSamplingThread, 512 | scaledSymbolWidth |
| QwtSeriesData, 552 | QwtPlotTradingCurve, 469 |
| QwtSeriesData, 552 QwtSeriesStore, 554 | scrollExtent |
| QwtSyntheticPointData, 590 | QwtAbstractLegend, 33 |
| sampleWidth | QwtLegend, 181 |
| QwtPlotAbstractBarChart, 288 | scrolledTo |
| samples | QwtAbstractSlider, 57 |
| QwtArraySeriesData, 68 | QwtDial, 122 |
| ScaleInterest | QwtKnob, 173 |
| QwtPlotItem, 352 | QwtSlider, 567 |
| ScaleSampleToCanvas | Second |
| QwtPlotAbstractBarChart, 286 | QwtDate, 101 |
| ScaleSamplesToAxes | SecondHand |
| QwtPlotAbstractBarChart, 286 | QwtAnalogClock, 64 |
| scaleChange | selected |
| QwtAbstractSlider, 57 | |
| QwtAbstractSlider, 57 QwtDial, 121 | QwtPicker, 244 QwtPlotPicker, 405 |
| | selection |
| QwtScaleWidget, 545 | |
| ScaleComponent QwtAbstractScaleDraw, 43 | QwtPicker, 244 |
| scaleDiv | SelectionType QwtPickerMachine, 257 |
| | |
| QwtAbstractScale, 38 | setAbortKey |
| QwtAbstractScaleDraw, 47 | QwtPanner, 232 |
| QwtPlotScaleItem, 432 scaleDraw | setAbstractScaleDraw QwtAbstractScale, 39 |
| | • |
| QwtDial, 121 QwtKnob, 173 | setAlarmBrush QwtThermo, 615 |
| QwtPlotScaleItem, 432 | setAlarmEnabled |
| | |
| QwtScaleWidget, 545, 546 QwtSlider, 567 | QwtThermo, 615 setAlarmLevel |
| QwtThermo, 615 | |
| | QwtThermo, 616 |
| scaleEngine | setAlignCanvasToScale QwtPlotLayout, 368 |
| QwtAbstractScale, 38 scaleInnerRect | setAlignCanvasToScales |
| QwtDial, 121 | QwtPlotLayout, 369 |
| scaleMap | setAlignment |
| • | QwtKnob, 173 |
| QwtAbstractScale, 38 QwtAbstractScaleDraw, 47, 48 | · · · · · · · · · · · · · · · · · · · |
| • • | QwtPlotLegendItem, 378 |
| scaleMaxMajor | QwtPlotScaleItem, 432 |
| QwtAbstractScale, 39 | QwtScaleDraw, 528 |
| scaleMaxMinor | QwtScaleWidget, 546 |
| QwtAbstractScale, 39 | setAlpha |
| ScalePosition | QwtPlotRasterItem, 411 |
| QwtSlider, 564 | setAngleRange |
| QwtThermo, 609 | QwtRoundScaleDraw, 510 |
| scalePosition | setAspectRatio |
| QwtSlider, 567 | QwtPlotRescaler, 425, 427 |
| QwtThermo, 615 | setAttribute |
| scaleRect | QwtPlotDirectPainter, 320 |
| QwtPlotItem, 356 | QwtScaleEngine, 533 |
| | |

| setAttributes | QwtText, 597 |
|------------------------------|----------------------------|
| QwtScaleEngine, 534 | setBorderRadius |
| setAutoDelete | QwtPlotCanvas, 300 |
| QwtPlotDict, 318 | QwtPlotLegendItem, 379 |
| setAutoReplot | QwtText, 597 |
| QwtPlot, 278 | setBorderWidth |
| setAxes | QwtKnob, 173 |
| QwtPlotItem, 358 | QwtSlider, 567 |
| setAxis | QwtThermo, 616 |
| QwtPlotPicker, 405 | QwtWheel, 629 |
| QwtPlotZoomer, 482 | setBoundingRect |
| setAxisAutoScale | QwtPointMapper, 493 |
| QwtPlot, 278 | setBrush |
| setAxisEnabled | QwtIntervalSymbol, 165 |
| QwtPlotMagnifier, 383 | QwtPlotCurve, 311 |
| QwtPlotPanner, 400 | QwtPlotHistogram, 339 |
| setAxisFont | QwtPlotIntervalCurve, 345 |
| QwtPlot, 279 | QwtPlotShapeItem, 441 |
| setAxisLabelAlignment | QwtPlotZoneItem, 476 |
| QwtPlot, 279 | QwtSymbol, 583 |
| setAxisLabelRotation | setCachePolicy |
| QwtPlot, 279 | QwtPlotRasterItem, 411 |
| setAxisMaxMajor | QwtSymbol, 583 |
| QwtPlot, 279 | setCanvas |
| setAxisMaxMinor | QwtPlot, 281 |
| QwtPlot, 280 | setCanvasBackground |
| setAxisScale | QwtPlot, 282 |
| QwtPlot, 280 | setCanvasMargin |
| setAxisScaleDiv | QwtPlotLayout, 369 |
| QwtPlot, 280 | setCanvasRect |
| setAxisScaleDraw | QwtPlotLayout, 369 |
| QwtPlot, 280 | setChecked |
| setAxisScaleEngine | QwtLegendLabel, 188 |
| QwtPlot, 281 | setChunkSize |
| setAxisTitle | QwtWeedingCurveFitter, 623 |
| QwtPlot, 281 | setClipRegion |
| setBackgroundBrush | QwtPlotDirectPainter, 321 |
| QwtPlotLegendItem, 378 | setClipping |
| QwtText, 596 | QwtPlotDirectPainter, 321 |
| setBackgroundMode | setColor |
| QwtPlotLegendItem, 379 | QwtAlphaColorMap, 62 |
| setBarTitles | QwtSymbol, 583 |
| QwtPlotMultiBarChart, 395 | QwtText, 597 |
| setBase | setColorBarEnabled |
| QwtScaleEngine, 534 | QwtScaleWidget, 546 |
| setBaseline | setColorBarWidth |
| QwtPlotAbstractBarChart, 288 | QwtScaleWidget, 546 |
| QwtPlotCurve, 311 | setColorInterval |
| QwtPlotHistogram, 339 | QwtLinearColorMap, 192 |
| setBorderDist | setColorMap |
| QwtScaleWidget, 546 | QwtPlotSpectroCurve, 447 |
| setBorderDistance | QwtPlotSpectrogram, 454 |
| QwtPlotLegendItem, 379 | QwtScaleWidget, 547 |
| QwtPlotScaleItem, 433 | QwtThermo, 616 |
| | setColorRange |
| setBorderFlags | |
| QwtInterval, 159 | QwtPlotSpectroCurve, 447 |
| setBorderPen | setCommands |
| QwtPlotLegendItem, 379 | QwtGraphic, 149 |

| setConrecFlag | setFrameShape |
|--|--|
| QwtPlotSpectrogram, 455 | QwtPlotGLCanvas, 326 |
| setContourLevels | setFrameStyle |
| QwtPlotSpectrogram, 455 | QwtColumnSymbol, 78 |
| setCursor | QwtPlotGLCanvas, 326 |
| QwtPanner, 232 | setGeometry |
| setCurveAttribute | QwtDynGridLayout, 133 |
| QwtPlotCurve, 312 | setGraphic |
| setCurveFitter | QwtSymbol, 584 |
| QwtPlotCurve, 312 | setGroove |
| setData | QwtSlider, 568 |
| QwtLegendLabel, 188 | setHand |
| QwtPlotSpectrogram, 455 | QwtAnalogClock, 67 |
| QwtSeriesStore, 554 | setHandleSize |
| setDateFormat | QwtSlider, 568 |
| QwtDateScaleDraw, 108 | setIcon |
| setDefaultContourPen | QwtLegendLabel, 188 |
| QwtPlotSpectrogram, 455, 456 | setIncSteps |
| setDefaultItemMode | QwtCounter, 93 |
| QwtLegend, 181 | setIndent |
| setDefaultSize | QwtTextLabel, 605 |
| QwtGraphic, 149 | setInterval |
| setDiscardFlag | QwtInterval, 159 |
| _ | |
| QwtPlotRenderer, 417 | QwtMatrixRasterData, 213 |
| setDiscardFlags | QwtPlotZoneItem, 476 |
| QwtPlotRenderer, 417 | QwtRasterData, 505 |
| setDisplayMode | QwtSamplingThread, 512 |
| QwtPlotSpectrogram, 456 | QwtScaleDiv, 519 |
| setEnabled | QwtSyntheticPointData, 590 |
| O | 1 - 1 - 1 - 1 1 1 1 1 1 |
| QwtMagnifier, 205 | setIntervalHint |
| QwtPanner, 232 | QwtPlotRescaler, 427 |
| QwtPanner, 232 QwtPicker, 244 | QwtPlotRescaler, 427 setInverted |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 | QwtPlotRescaler, 427 setInverted QwtWheel, 630 |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotItem, 358 |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotItem, 358 setItemMargin |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotItem, 358 setItemMargin QwtPlotLegendItem, 380 |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 setFlag | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotItem, 358 setItemMargin QwtPlotLegendItem, 380 setItemMode |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 setFlag QwtPointMapper, 493 | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotItem, 358 setItemMargin QwtPlotLegendItem, 380 setItemMode QwtLegendLabel, 189 |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 setFlag QwtPointMapper, 493 setFlags | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotItem, 358 setItemMargin QwtPlotLegendItem, 380 setItemMode QwtLegendLabel, 189 setItemSpacing |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 setFlag QwtPointMapper, 493 | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotItem, 358 setItemMargin QwtPlotLegendItem, 380 setItemMode QwtLegendLabel, 189 |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 setFlag QwtPointMapper, 493 setFlags | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotItem, 358 setItemMargin QwtPlotLegendItem, 380 setItemMode QwtLegendLabel, 189 setItemSpacing |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 setFlag QwtPointMapper, 493 setFlags QwtPointMapper, 493 | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotItem, 358 setItemMargin QwtPlotLegendItem, 380 setItemMode QwtLegendLabel, 189 setItemSpacing QwtPlotLegendItem, 380 |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 setFlag QwtPointMapper, 493 setFlags QwtPointMapper, 493 setFocusIndicator | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotItem, 358 setItemMargin QwtPlotLegendItem, 380 setItemMode QwtLegendLabel, 189 setItemSpacing QwtPlotLegendItem, 380 setKeyFactor |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 setFlag QwtPointMapper, 493 setFlags QwtPointMapper, 493 setFocusIndicator QwtPlotCanvas, 301 | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotItem, 358 setItemMargin QwtPlotLegendItem, 380 setItemMode QwtLegendLabel, 189 setItemSpacing QwtPlotLegendItem, 380 setKeyFactor QwtMagnifier, 205 |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 setFlag QwtPointMapper, 493 setFlags QwtPointMapper, 493 setFocusIndicator QwtPlotCanvas, 301 setFont | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotItem, 358 setItemMargin QwtPlotLegendItem, 380 setItemMode QwtLegendLabel, 189 setItemSpacing QwtPlotLegendItem, 380 setKeyFactor QwtMagnifier, 205 setKeyPattern |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 setFlag QwtPointMapper, 493 setFlags QwtPointMapper, 493 setFocusIndicator QwtPlotCanvas, 301 setFont QwtPlotLegendItem, 379 | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotItem, 358 setItemMargin QwtPlotLegendItem, 380 setItemMode QwtLegendLabel, 189 setItemSpacing QwtPlotLegendItem, 380 setKeyFactor QwtMagnifier, 205 setKeyPattern QwtEventPattern, 139 |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 setFlag QwtPointMapper, 493 setFlags QwtPointMapper, 493 setFocusIndicator QwtPlotCanvas, 301 setFont QwtPlotScaleItem, 433 | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotItem, 358 setItemMargin QwtPlotLegendItem, 380 setItemMode QwtLegendLabel, 189 setItemSpacing QwtPlotLegendItem, 380 setKeyFactor QwtMagnifier, 205 setKeyPattern QwtEventPattern, 139 setKnobStyle |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 setFlag QwtPointMapper, 493 setFlags QwtPointMapper, 493 setFocusIndicator QwtPlotCanvas, 301 setFont QwtPlotScaleItem, 433 QwtText, 597 | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotItem, 358 setItemMargin QwtPlotLegendItem, 380 setItemMode QwtLegendLabel, 189 setItemSpacing QwtPlotLegendItem, 380 setKeyFactor QwtMagnifier, 205 setKeyPattern QwtEventPattern, 139 setKnobStyle QwtKnob, 174 |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 setFlag QwtPointMapper, 493 setFlags QwtPointMapper, 493 setFocusIndicator QwtPlotCanvas, 301 setFont QwtPlotScaleItem, 433 QwtText, 597 setFooter | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotItem, 358 setItemMargin QwtPlotLegendItem, 380 setItemMode QwtLegendLabel, 189 setItemSpacing QwtPlotLegendItem, 380 setKeyFactor QwtMagnifier, 205 setKeyPattern QwtEventPattern, 139 setKnobStyle QwtKnob, 174 setKnobWidth |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 setFlag QwtPointMapper, 493 setFlags QwtPointMapper, 493 setFocusIndicator QwtPlotCanvas, 301 setFont QwtPlotScaleItem, 433 QwtText, 597 setFooter QwtPlot, 282 | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotLegendItem, 380 setItemMargin QwtPlotLegendItem, 380 setItemSpacing QwtLegendLabel, 189 setItemSpacing QwtPlotLegendItem, 380 setKeyFactor QwtMagnifier, 205 setKeyPattern QwtEventPattern, 139 setKnobStyle QwtKnob, 174 setKnobWidth QwtKnob, 174 |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 setFlag QwtPointMapper, 493 setFlags QwtPointMapper, 493 setFocusIndicator QwtPlotCanvas, 301 setFont QwtPlotScaleItem, 379 QwtPlotScaleItem, 433 QwtText, 597 setFooter QwtPlot, 282 setFooterRect | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotLegendItem, 380 setItemMargin QwtPlotLegendItem, 380 setItemSpacing QwtLegendLabel, 189 setItemSpacing QwtPlotLegendItem, 380 setKeyFactor QwtMagnifier, 205 setKeyPattern QwtEventPattern, 139 setKnobStyle QwtKnob, 174 setKnobWidth QwtKnob, 174 setLabel |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 setFlag QwtPointMapper, 493 setFlags QwtPointMapper, 493 setFocusIndicator QwtPlotCanvas, 301 setFont QwtPlotScaleItem, 379 QwtPlotScaleItem, 433 QwtText, 597 setFooter QwtPlot, 282 setFooterRect QwtPlotLayout, 369 | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotLegendItem, 380 setItemMargin QwtPlotLegendItem, 380 setItemSpacing QwtPlotLegendItem, 380 setKeyFactor QwtMagnifier, 205 setKeyPattern QwtEventPattern, 139 setKnobStyle QwtKnob, 174 setKnobWidth QwtKnob, 174 setLabel QwtPlotMarker, 388 |
| QwtPanner, 232 QwtPicker, 244 QwtPlotRescaler, 427 setExpandingDirection QwtPlotRescaler, 427 setExpandingDirections QwtDynGridLayout, 132 setFillBrush QwtThermo, 616 setFitMode QwtSplineCurveFitter, 575 setFlag QwtPointMapper, 493 setFlags QwtPointMapper, 493 setFocusIndicator QwtPlotCanvas, 301 setFont QwtPlotScaleItem, 433 QwtText, 597 setFooter QwtPlot, 282 setFooterRect QwtPlotLayout, 369 setFrameShadow | QwtPlotRescaler, 427 setInverted QwtWheel, 630 setInvertedControls QwtAbstractSlider, 57 setItemAttribute QwtPlotItem, 358 setItemInterest QwtPlotLegendItem, 380 setItemMargin QwtPlotLegendItem, 380 setItemSpacing QwtPlotLegendItem, 380 setKeyFactor QwtMagnifier, 205 setKeyPattern QwtEventPattern, 139 setKnobStyle QwtKnob, 174 setKnobWidth QwtKnob, 174 setLabel QwtPlotMarker, 388 setLabelAlignment |

| QwtScaleWidget, 547 | QwtKnob, 174 |
|--------------------------------------|--|
| setLabelMap | setMaskMode |
| QwtCompassScaleDraw, 86 | QwtWidgetOverlay, 641 |
| setLabelOrientation | setMass |
| QwtPlotMarker, 388 | QwtWheel, 630 |
| setLabelRotation | setMaxColumns |
| QwtScaleDraw, 529 | QwtDynGridLayout, 133 |
| QwtScaleWidget, 547 | QwtLegend, 182 |
| setLayoutAttribute | QwtPlotLegendItem, 380 |
| QwtText, 597 | setMaxScaleArc |
| setLayoutFlag | QwtDial, 122 |
| QwtPlotRenderer, 419 | setMaxStackDepth |
| QwtScaleWidget, 547 | QwtPlotZoomer, 482 |
| setLayoutFlags | setMaxSymbolWidth |
| QwtPlotRenderer, 419 | QwtPlotTradingCurve, 469 |
| setLayoutHint | setMaxValue |
| QwtPlotAbstractBarChart, 288 | QwtInterval, 159 |
| setLayoutPolicy | setMaxWeeks |
| QwtPlotAbstractBarChart, 288 | QwtDateScaleEngine, 113 |
| setLegendAttribute | setMaximum |
| QwtPlotCurve, 312 | QwtCounter, 93 |
| setLegendIconSize | QwtWheel, 630 |
| QwtPlotItem, 358 | setMidLineWidth |
| setLegendMode | QwtPlotGLCanvas, 326 |
| QwtPlotBarChart, 294 | setMinBorderDist |
| QwtPlotShapeItem, 441 | QwtScaleWidget, 548 |
| setLegendPosition | setMinScaleArc |
| QwtPlotLayout, 369, 371 | QwtDial, 123 |
| setLegendRatio | setMinSymbolWidth |
| QwtPlotLayout, 371 | QwtPlotTradingCurve, 471 |
| setLegendRect | setMinValue |
| QwtPlotLayout, 371 | QwtInterval, 159 |
| setLength | setMinimum |
| QwtScaleDraw, 529 | QwtCounter, 93 |
| setLinePen | QwtWheel, 630 |
| QwtPlotMarker, 388, 389 | setMinimumExtent |
| setLineStyle | QwtAbstractScaleDraw, 48 |
| QwtPlotMarker, 389 | setMinorPen |
| setLineWidth | QwtPlotGrid, 330 |
| QwtColumnSymbol, 79 | setMode |
| QwtDial, 122 | QwtDial, 123 |
| QwtPlotGLCanvas, 326 | QwtLinearColorMap, 193 |
| setLowerBound | QwtNullPaintDevice, 217 |
| QwtAbstractScale, 39 | setMouseButton |
| QwtScaleDiv, 519 | QwtMagnifier, 205 |
| setMajorPen | QwtPanner, 232 |
| QwtPlotGrid, 329, 330 | setMouseFactor |
| setMargin | QwtMagnifier, 205 |
| QwtPlotAbstractBarChart, 289 | setMousePattern |
| QwtPlotLegendItem, 380 | QwtEventPattern, 139 |
| QwtPlotTextLabel, 462 | setNeedle |
| QwtScaleWidget, 547 | QwtDial, 123 |
| QwtTextLabel, 605 | setNumButtons |
| setMargins | QwtCounter, 93 |
| - | setNumThornLevels |
| QwtScaleEngine, 534 setMarkerSize | |
| | QwtSimpleCompassRose, 561 setNumThorns |
| QwtKnob, 174 | |
| setMarkerStyle | QwtSimpleCompassRose, 561 |

| IN T | IDI II |
|---------------------------|----------------------------|
| setNumTurns | setPlotLayout |
| QwtKnob, 174 | QwtPlot, 282 |
| setOrientation | setPoint |
| QwtPlotSeriesItem, 437 | QwtPointPolar, 498 |
| QwtPlotZoneItem, 477 | setPoints |
| QwtSlider, 568 | QwtSpline, 573 |
| QwtThermo, 617 | setPolygon |
| QwtWheel, 631 | QwtPlotShapeItem, 442 |
| setOrientations | setPolylineSplitting |
| QwtPanner, 232 | QwtPainter, 225 |
| setOrigin | setPosition |
| QwtDial, 123 | QwtPlotScaleItem, 433 |
| QwtThermo, 617 | setRadius |
| setOriginMode | QwtRoundScaleDraw, 510 |
| QwtThermo, 617 | setRange |
| setPageStepCount | QwtCounter, 94 |
| QwtWheel, 631 | QwtWheel, 631 |
| setPageSteps | setRangeFlags |
| QwtAbstractSlider, 57 | QwtThermo, 617 |
| setPaintAttribute | setRawSamples |
| QwtPlotCanvas, 301 | QwtPlotCurve, 313 |
| QwtPlotCurve, 312 | setReadOnly |
| QwtPlotIntervalCurve, 345 | QwtAbstractSlider, 58 |
| QwtPlotRasterItem, 411 | |
| | QwtCounter, 94 setRect |
| QwtPlotShapeItem, 442 | |
| QwtPlotSpectroCurve, 447 | QwtPixelMatrix, 261 |
| QwtPlotTradingCurve, 471 | QwtPlotShapeItem, 443 |
| QwtText, 598 | setRectOfInterest |
| setPaintInterval | QwtAbstractSeriesStore, 51 |
| QwtScaleMap, 538 | QwtSeriesData, 552 |
| setPalette | QwtSeriesStore, 554 |
| QwtColumnSymbol, 79 | QwtSyntheticPointData, 590 |
| QwtDialNeedle, 126 | setReference |
| QwtPlotScaleItem, 433 | QwtScaleEngine, 534 |
| setPath | setReferenceAxis |
| QwtSymbol, 584 | QwtPlotRescaler, 429 |
| setPen | setRenderFlags |
| QwtIntervalSymbol, 165 | QwtText, 598 |
| QwtPlotCurve, 312, 313 | setRenderHint |
| QwtPlotGrid, 330, 331 | QwtGraphic, 149 |
| QwtPlotHistogram, 339 | QwtPlotItem, 359 |
| QwtPlotIntervalCurve, 346 | setRenderMode |
| QwtPlotShapeItem, 442 | QwtWidgetOverlay, 641 |
| QwtPlotZoneItem, 477 | setRenderThreadCount |
| QwtSymbol, 585 | QwtPlotItem, 359 |
| setPenWidth | setRenderTolerance |
| QwtAbstractScaleDraw, 48 | QwtPlotShapeItem, 443 |
| QwtPlotSpectroCurve, 447 | setResampleMode |
| setPinPoint | QwtMatrixRasterData, 213 |
| QwtSymbol, 585 | setRescalePolicy |
| setPinPointEnabled | QwtPlotRescaler, 429 |
| QwtSymbol, 585 | setResizeMode |
| setPipeWidth | QwtPicker, 244 |
| QwtThermo, 617 | setRose |
| setPixmap | QwtCompass, 82 |
| QwtSymbol, 586 | setRoundingAlignment |
| setPlainText | QwtPainter, 225 |
| QwtTextLabel, 605 | setRubberBand |
| antionization, ovo | 551. (6555. 54.14 |

| QwtPicker, 244 | QwtLegendLabel, 189 |
|-------------------------------|------------------------------|
| setRubberBandPen | QwtPlotAbstractBarChart, 289 |
| QwtPicker, 245 | QwtPlotLayout, 372 |
| setSamples | QwtPlotLegendItem, 380 |
| QwtArraySeriesData, 68 | QwtPlotMarker, 389 |
| QwtPlotBarChart, 294, 295 | QwtScaleWidget, 548 |
| QwtPlotCurve, 313, 314 | QwtSlider, 569 |
| QwtPlotHistogram, 340 | QwtThermo, 618 |
| QwtPlotIntervalCurve, 346 | setSpline |
| QwtPlotMultiBarChart, 395 | QwtSplineCurveFitter, 575 |
| QwtPlotSpectroCurve, 447, 448 | setSplineSize |
| QwtPlotTradingCurve, 471 | QwtSplineCurveFitter, 576 |
| setScale | setSplineType |
| QwtAbstractScale, 39, 40 | QwtSpline, 573 |
| setScaleArc | setStateMachine |
| QwtDial, 123 | QwtPicker, 245 |
| setScaleDiv | setStepAlignment |
| QwtAbstractScaleDraw, 48 | QwtAbstractSlider, 58 |
| QwtPlotScaleItem, 434 | QwtWheel, 632 |
| QwtScaleWidget, 548 | setStepButton1 |
| setScaleDivFromAxis | QwtCounter, 94 |
| QwtPlotScaleItem, 434 | setStepButton2 |
| setScaleDraw | QwtCounter, 94 |
| QwtDial, 124 | setStepButton3 |
| QwtKnob, 175 | QwtCounter, 95 |
| QwtPlotScaleItem, 434 | setStyle |
| QwtScaleWidget, 548 | QwtColumnSymbol, 79 |
| QwtSlider, 568 | QwtIntervalSymbol, 166 |
| QwtThermo, 618 | QwtPlotCurve, 314 |
| setScaleEngine | QwtPlotHistogram, 340 |
| QwtAbstractScale, 40 | QwtPlotIntervalCurve, 346 |
| setScaleInterval | QwtPlotMultiBarChart, 396 |
| QwtScaleMap, 539 | QwtSymbol, 586 |
| setScaleMaxMajor | setSvgDocument |
| QwtAbstractScale, 40 | QwtSymbol, 587 |
| setScaleMaxMinor | setSymbol |
| QwtAbstractScale, 41 | QwtPlotBarChart, 295 |
| setScalePosition | QwtPlotCurve, 314 |
| QwtSlider, 569 | QwtPlotHistogram, 340 |
| QwtThermo, 618 | QwtPlotIntervalCurve, 348 |
| setScaleRect | QwtPlotMarker, 389 |
| QwtPlotLayout, 371 | QwtPlotMultiBarChart, 396 |
| setScaleStepSize | setSymbolBrush |
| QwtAbstractScale, 41 | QwtPlotTradingCurve, 471 |
| setShape | setSymbolExtent |
| QwtPlotShapeItem, 443 | QwtPlotTradingCurve, 472 |
| setShrinkFactor | setSymbolPen |
| QwtSimpleCompassRose, 561 | QwtPlotTradingCurve, 472 |
| setSingleStep | setSymbolStyle |
| QwtCounter, 94 | QwtPlotTradingCurve, 472 |
| QwtWheel, 631 | setText |
| setSingleSteps | QwtLegendLabel, 189 |
| QwtAbstractSlider, 58 | QwtPlotTextLabel, 462 |
| setSize | QwtText, 598 |
| QwtSymbol, 586 | QwtTextLabel, 605, 607 |
| QwtSyntheticPointData, 590 | setTextEngine |
| setSpacing | QwtText, 598 |
| QwtAbstractScaleDraw, 48 | setTextPen |
| GWW Dollactocale Diaw, 70 | JOLIOALI GII |

| QwtPlotLegendItem, 381 | QwtThermo, 618 |
|----------------------------|---------------------------|
| setTickCount | QwtWheel, 633 |
| QwtWheel, 632 | setValueMatrix |
| setTickLength | QwtMatrixRasterData, 213 |
| QwtAbstractScaleDraw, 48 | setValues |
| setTicks | QwtLegendData, 184 |
| QwtScaleDiv, 519 | setViewAngle |
| setTime | QwtWheel, 633 |
| QwtAnalogClock, 67 | setVisible |
| setTimeSpec | QwtPlotItem, 359 |
| QwtDateScaleDraw, 108 | setWeek0Type |
| QwtDateScaleEngine, 113 | QwtDateScaleDraw, 108 |
| setTitle | QwtDateScaleEngine, 113 |
| QwtPlot, 282, 283 | setWheelBorderWidth |
| QwtPlotItem, 359 | QwtWheel, 633 |
| QwtScaleWidget, 549 | setWheelFactor |
| setTitleRect | QwtMagnifier, 205 |
| QwtPlotLayout, 372 | setWheelModifiers |
| setTolerance | QwtMagnifier, 206 |
| QwtWeedingCurveFitter, 624 | setWheelWidth |
| setTotalAngle | QwtWheel, 634 |
| QwtKnob, 175 | setWidth |
| QwtWheel, 632 | QwtDialSimpleNeedle, 128 |
| setTotalSteps | QwtIntervalSymbol, 166 |
| QwtAbstractSlider, 58 | QwtSimpleCompassRose, 561 |
| setTrackerFont | setWrapping |
| QwtPicker, 245 | QwtAbstractSlider, 59 |
| setTrackerMode | QwtCounter, 95 |
| QwtPicker, 245 | QwtWheel, 634 |
| setTrackerPen | setXAxis |
| QwtPicker, 245 | QwtPlotItem, 360 |
| setTracking | setXDiv |
| QwtAbstractSlider, 59 | QwtPlotGrid, 331 |
| QwtWheel, 632 | setYAxis |
| setTransformation | QwtPlotItem, 360 |
| QwtAbstractScaleDraw, 50 | setYDiv |
| QwtScaleEngine, 535 | QwtPlotGrid, 331 |
| QwtScaleMap, 539 | setZ |
| QwtScaleWidget, 549 | QwtPlotItem, 360 |
| setTrough | setZoomBase |
| QwtSlider, 569 | QwtPlotZoomer, 482, 484 |
| setUpdateInterval | setZoomInKey |
| QwtSlider, 569 | QwtMagnifier, 206 |
| QwtWheel, 633 | setZoomOutKey |
| setUpperBound | QwtMagnifier, 206 |
| QwtAbstractScale, 41 | setZoomStack |
| | |
| QwtScaleDiv, 520 | QwtPlotZoomer, 484 |
| setUtcOffset | Shadow |
| QwtDateScaleDraw, 108 | QwtDial, 117 |
| QwtDateScaleEngine, 113 | QwtPlotGLCanvas, 323 |
| setValid | Shape |
| QwtAbstractSlider, 59 | QwtPlotGLCanvas, 323 |
| QwtCounter, 95 | shape |
| setValue | QwtPlotShapeItem, 443 |
| QwtAbstractSlider, 59 | shrinkFactor |
| QwtCounter, 95 | QwtSimpleCompassRose, 562 |
| QwtLegendData, 184 | singleStep |
| QwtMatrixRasterData, 213 | QwtCounter, 95 |

| 0 114/1 004 | 01 0 |
|------------------------------|----------------------------|
| QwtWheel, 634 | Star2 |
| singleSteps | QwtSymbol, 579 |
| QwtAbstractSlider, 59 | start |
| size | QwtSystemClock, 592 |
| QwtArraySeriesData, 69 | startBorderDist |
| QwtCPointerData, 98 | QwtScaleWidget, 549 |
| QwtPointArrayData, 491 | State |
| QwtSeriesData, 552 | QwtPainterCommand, 226 |
| QwtSymbol, 587 | stateData |
| QwtSyntheticPointData, 591 | QwtPainterCommand, 228 |
| sizeHint | stateMachine |
| QwtArrowButton, 71 | QwtPicker, 247 |
| QwtDial, 124 | stepAlignment |
| QwtDynGridLayout, 133 | QwtAbstractSlider, 60 |
| QwtKnob, 175 | QwtWheel, 634 |
| | |
| QwtPlot, 283 | Steps |
| QwtScaleWidget, 549 | QwtPlotCurve, 305 |
| QwtSlider, 570 | Sticks |
| QwtThermo, 618 | QwtPlotCurve, 305 |
| QwtWheel, 634 | stop |
| sizeMetrics | QwtSamplingThread, 513 |
| QwtGraphic, 150 | Stretch |
| QwtNullPaintDevice, 217 | QwtPicker, 238 |
| sliderMoved | stretchGrid |
| QwtAbstractSlider, 60 | QwtDynGridLayout, 133 |
| sliderPressed | stretchSelection |
| QwtAbstractSlider, 60 | QwtPicker, 247 |
| sliderRect | strip |
| | |
| QwtSlider, 570 | QwtScaleEngine, 535 |
| sliderReleased | Style |
| QwtAbstractSlider, 60 | QwtColumnSymbol, 77 |
| spacing | QwtCompassMagnetNeedle, 83 |
| QwtAbstractScaleDraw, 50 | QwtCompassWindArrow, 88 |
| QwtLegendLabel, 189 | QwtDialSimpleNeedle, 127 |
| QwtPlotAbstractBarChart, 289 | QwtIntervalSymbol, 164 |
| QwtPlotLayout, 372 | QwtSymbol, 578 |
| QwtPlotLegendItem, 381 | style |
| QwtPlotMarker, 389 | QwtColumnSymbol, 79 |
| QwtScaleWidget, 549 | QwtIntervalSymbol, 166 |
| QwtSlider, 570 | QwtPlotCurve, 315 |
| QwtThermo, 619 | QwtPlotHistogram, 340 |
| specialSymbol | QwtPlotIntervalCurve, 348 |
| QwtPlotBarChart, 295 | QwtPlotMultiBarChart, 396 |
| QwtPlotMultiBarChart, 396 | QwtSymbol, 587 |
| Spline | |
| · | Style1 |
| QwtSplineCurveFitter, 575 | QwtCompassWindArrow, 88 |
| spline | Style2 |
| QwtSplineCurveFitter, 576 | QwtCompassWindArrow, 88 |
| splineSize | Styled |
| QwtSplineCurveFitter, 576 | QwtKnob, 169 |
| SplineType | Sunken |
| QwtSpline, 572 | QwtDial, 117 |
| splineType | QwtKnob, 169 |
| QwtSpline, 573 | QwtPlotGLCanvas, 323 |
| Stacked | SvgDocument |
| QwtPlotMultiBarChart, 392 | QwtSymbol, 579 |
| Star1 | swapData |
| QwtSymbol, 579 | QwtSeriesStore, 555 |
| CAMICALIDOI: 7/7 | WILDERIESOLUIE, JJJ |

| symbol | QwtPlotShapeItem, 443 |
|---------------------------|----------------------------|
| QwtPlotBarChart, 296 | QwtPlotSpectroCurve, 448 |
| QwtPlotCurve, 315 | QwtPlotTradingCurve, 473 |
| QwtPlotHistogram, 341 | QwtText, 599 |
| QwtPlotIntervalCurve, 348 | testPixel |
| QwtPlotMarker, 390 | QwtPixelMatrix, 261 |
| QwtPlotMultiBarChart, 397 | testRenderHint |
| symbolBrush | QwtGraphic, 150 |
| QwtPlotTradingCurve, 473 | QwtPlotItem, 362 |
| symbolExtent | text |
| QwtPlotTradingCurve, 473 | QwtPlotTextLabel, 462 |
| symbolPen | QwtText, 599 |
| QwtPlotTradingCurve, 473 | textEngine |
| SymbolStyle | _ |
| QwtPlotTradingCurve, 466 | QwtText, 599, 600 |
| symbolStyle | TextFormat |
| QwtPlotTradingCurve, 473 | QwtText, 594 |
| Symmetric Symmetric | textMargins |
| | QwtMathMLTextEngine, 210 |
| QwtScaleEngine, 531 | QwtPlainTextEngine, 263 |
| symmetrize | QwtRichTextEngine, 507 |
| QwtInterval, 160 | QwtTextEngine, 602 |
| syncScale | textPen |
| QwtPlotRescaler, 429 | QwtPlotLegendItem, 381 |
| takeAt | textRect |
| QwtDynGridLayout, 133 | QwtPlotTextLabel, 462 |
| TeXText | QwtTextLabel, 607 |
| QwtText, 595 | textSize |
| testAndSetPixel | QwtMathMLTextEngine, 210 |
| QwtPixelMatrix, 261 | QwtPlainTextEngine, 263 |
| testAttribute | QwtRichTextEngine, 507 |
| | QwtText, 600 |
| QwtPlotDirectPainter, 321 | QwtTextEngine, 603 |
| QwtScaleEngine, 535 | ThinStyle |
| testConrecFlag | QwtCompassMagnetNeedle, 83 |
| QwtPlotSpectrogram, 456 | Tick |
| testCurveAttribute | QwtKnob, 169 |
| QwtPlotCurve, 315 | tickCount |
| testDiscardFlag | QwtWheel, 634 |
| QwtPlotRenderer, 419 | tickLabel |
| testDisplayMode | QwtAbstractScaleDraw, 50 |
| QwtPlotSpectrogram, 456 | tickLength |
| testFlag | - |
| QwtPointMapper, 493 | QwtAbstractScaleDraw, 50 |
| testItemAttribute | TickType |
| QwtPlotItem, 360 | QwtScaleDiv, 515 |
| testItemInterest | Ticks |
| QwtPlotItem, 360 | QwtAbstractScaleDraw, 44 |
| testLayoutAttribute | ticks |
| QwtText, 599 | QwtScaleDiv, 520 |
| testLayoutFlag | time |
| QwtPlotRenderer, 419 | QwtOHLCSample, 220 |
| QwtScaleWidget, 550 | timeSpec |
| testLegendAttribute | QwtDateScaleDraw, 109 |
| QwtPlotCurve, 315 | QwtDateScaleEngine, 114 |
| testPaintAttribute | timerEvent |
| QwtPlotCanvas, 301 | QwtSlider, 570 |
| QwtPlotCurve, 315 | QwtWheel, 635 |
| QwtPlotIntervalCurve, 348 | title |
| QwtPlotRasterItem, 412 | QwtLegendData, 184 |
| | |

| QwtPlot, 283 | trackerPosition |
|-------------------------------|----------------------------|
| QwtPlotItem, 362 | QwtPicker, 248 |
| QwtScaleWidget, 550 | trackerRect |
| TitleInverted | QwtPicker, 248 |
| QwtScaleWidget, 542 | trackerText |
| titleHeightForWidth | QwtPicker, 248 |
| QwtScaleWidget, 550 | QwtPlotPicker, 406 |
| titleLabel | trackerTextF |
| QwtPlot, 283 | QwtPlotPicker, 406 |
| titleRect | TrailingScale |
| QwtPlotLayout, 372 | QwtSlider, 564 |
| toDateTime | QwtThermo, 610 |
| QwtDate, 103 | transform |
| QwtDateScaleDraw, 109 | QwtAbstractScale, 41 |
| QwtDateScaleEngine, 114 | QwtLogTransform, 200 |
| toDouble | QwtNullTransform, 218 |
| QwtDate, 103 | QwtPlot, 283 |
| tolmage | QwtPlotPicker, 406 |
| QwtGraphic, 150 | QwtPowerTransform, 502 |
| QwtPointMapper, 494 | QwtScaleMap, 539 |
| toPixmap | QwtTransform, 621 |
| QwtGraphic, 151 | transformation |
| toPoint | QwtScaleEngine, 535 |
| QwtPoint3D, 487 | TransformationFlag |
| QwtPointPolar, 498 | QwtPointMapper, 492 |
| toPoints | TransformationFlags |
| QwtPointMapper, 494 | QwtPointMapper, 492 |
| toPointsF | transition |
| QwtPointMapper, 494 | QwtPicker, 248 |
| toPolygon | Triangle |
| QwtPointMapper, 495 | QwtKnob, 169 |
| toPolygonF | QwtSymbol, 579 |
| QwtPointMapper, 495 | TriangleStyle |
| toRect | QwtCompassMagnetNeedle, 83 |
| QwtColumnRect, 76 | Tube |
| toString | QwtPlotIntervalCurve, 343 |
| QwtDate, 103 | Type |
| tolerance | QwtPainterCommand, 226 |
| QwtWeedingCurveFitter, 624 | type |
| TopLegend | QwtPainterCommand, 228 |
| • • | |
| QwtPlot, 268 | UTriangle |
| TopScale | QwtSymbol, 579 |
| QwtScaleDraw, 522 TopToBottom | updateAxes |
| • | QwtPlot, 283 |
| QwtColumnRect, 76 | updateCanvasMargins |
| totalAngle | QwtPlot, 284 |
| QwtKnob, 175 | updateInterval |
| QwtWheel, 635 | QwtSlider, 570 |
| totalSteps | QwtWheel, 635 |
| QwtAbstractSlider, 60 | updateLayout |
| trackerFont | QwtPlot, 284 |
| QwtPicker, 247 | updateLegend |
| trackerMode | QwtAbstractLegend, 34 |
| QwtPicker, 247 | QwtLegend, 182 |
| trackerOverlay | QwtPlot, 284 |
| QwtPicker, 247 | QwtPlotItem, 362 |
| trackerPen | QwtPlotLegendItem, 381 |
| QwtPicker, 248 | updateOverlay |
| | |

| QwtWidgetOverlay, 642 | QwtLegendData, 186 |
|---------------------------------------|-----------------------------|
| updateScaleDiv | verticalScrollBar |
| QwtPlotGrid, 331 | QwtLegend, 182 |
| QwtPlotItem, 362 | viewAngle |
| QwtPlotScaleItem, 434 | QwtWheel, 636 |
| QwtPlotSeriesItem, 437 | viewBox |
| updateScales | QwtPlotSvgItem, 459 |
| QwtPlotRescaler, 429 | |
| updateState | WeedOutPoints |
| QwtGraphic, 151 | QwtPointMapper, 492 |
| updateWidget | Week |
| QwtLegend, 182 | QwtDate, 101 |
| upperBound | Week0Type |
| QwtAbstractScale, 42 | QwtDate, 101 |
| QwtScaleDiv, 520 | week0Type |
| upperMargin | QwtDateScaleDraw, 109 |
| QwtScaleEngine, 535 | QwtDateScaleEngine, 114 |
| usedColor | weekNumber |
| QwtText, 600 | QwtDate, 104 |
| usedFont | wheelBorderWidth |
| QwtText, 600 | QwtWheel, 636 |
| UserCurve | wheelEvent |
| QwtPlotCurve, 305 | QwtAbstractSlider, 61 |
| QwtPlotIntervalCurve, 343 | QwtCounter, 96 |
| UserRubberBand | QwtDial, 124 |
| QwtPicker, 238 | QwtWheel, 636 |
| UserStyle | wheelFactor |
| QwtColumnSymbol, 77 | QwtMagnifier, 206 |
| QwtPlotHistogram, 334 | wheelModifiers |
| G . | QwtMagnifier, 207 |
| QwtSymbol, 579 | wheelMoved |
| UserSymbol OutlintoryalSymbol 164 | QwtWheel, 636 |
| QwtIntervalSymbol, 164 | wheelPressed |
| QwtPlotTradingCurve, 466 utcOffset | QwtWheel, 637 |
| | wheelRect |
| QwtDate, 104 | QwtWheel, 637 |
| QwtDateScaleDraw, 109 | wheelReleased |
| QwtDateScaleEngine, 114 | QwtWheel, 637 |
| VLine | wheelWidth |
| QwtPlotMarker, 385 | QwtWheel, 637 |
| QwtSymbol, 579 | widgetEnterEvent |
| VLineRubberBand | QwtPicker, 249 |
| QwtPicker, 238 | widgetKeyPressEvent |
| value | QwtMagnifier, 207 |
| QwtCounter, 96 | QwtPanner, 233 |
| QwtLegendData, 184 | QwtPicker, 249 |
| QwtMatrixRasterData, 214 | QwtPlotZoomer, 484 |
| QwtRasterData, 505 | widgetKeyReleaseEvent |
| QwtSpline, 573 | QwtMagnifier, 207 |
| QwtWheel, 635 | QwtPanner, 233 |
| valueAt | QwtPicker, 249 |
| QwtWheel, 635 | widgetLeaveEvent |
| valueChanged | QwtPicker, 249 |
| QwtAbstractSlider, 60 | widgetMouseDoubleClickEvent |
| | QwtPicker, 249 |
| QwtCounter, 96 | |
| QwtWheel, 636 | widgetMouseMoveEvent |
| valueMatrix | QwtMagnifier, 207 |
| QwtMatrixRasterData, 214 | QwtPanner, 233 |
| values | QwtPicker, 251 |

| widgetMousePressEvent | Z |
|----------------------------|--------------------|
| QwtMagnifier, 207 | QwtPlotItem, 363 |
| QwtPanner, 233 | QwtPoint3D, 488 |
| QwtPicker, 251 | zoom |
| widgetMouseReleaseEvent | QwtPlotZoomer, 485 |
| QwtMagnifier, 208 | zoomBase |
| QwtPanner, 233 | QwtPlotZoomer, 485 |
| QwtPicker, 251 | zoomRect |
| QwtPlotZoomer, 484 | QwtPlotZoomer, 485 |
| widgetWheelEvent | zoomRectIndex |
| QwtMagnifier, 208 | QwtPlotZoomer, 485 |
| QwtPicker, 251 | zoomStack |
| | |
| width | QwtPlotZoomer, 486 |
| QwtDialSimpleNeedle, 128 | zoomed |
| QwtInterval, 160 | QwtPlotZoomer, 485 |
| QwtIntervalSymbol, 166 | |
| QwtSimpleCompassRose, 562 | |
| wrapping | |
| QwtAbstractSlider, 61 | |
| QwtCounter, 96 | |
| QwtWheel, 637 | |
| | |
| X | |
| QwtPoint3D, 488 | |
| QwtSyntheticPointData, 591 | |
| xBottom | |
| QwtPlot, 267 | |
| XCross | |
| QwtSymbol, 579 | |
| хТор | |
| QwtPlot, 267 | |
| xData | |
| QwtCPointerData, 98 | |
| QwtPointArrayData, 491 | |
| xEnabled | |
| QwtPlotGrid, 331 | |
| xMinEnabled | |
| | |
| QwtPlotGrid, 332 | |
| xScaleDiv | |
| QwtPlotGrid, 332 | |
| у | |
| QwtPoint3D, 488 | |
| QwtSyntheticPointData, 591 | |
| yLeft | |
| QwtPlot, 267 | |
| | |
| yRight 007 | |
| QwtPlot, 267 | |
| yData | |
| QwtCPointerData, 98 | |
| QwtPointArrayData, 491 | |
| yEnabled | |
| QwtPlotGrid, 332 | |
| yMinEnabled | |
| QwtPlotGrid, 332 | |
| yScaleDiv | |
| QwtPlotGrid, 332 | |
| Year | |
| QwtDate, 101 | |