Package ‘qtpaint’

November 11, 2015

Version 0.9.1
Title Qt-Based Painting Infrastructure
Author Michael Lawrence, Deepayan Sarkar
Depends R (>= 2.6.0), utils, qtbase (>= 0.99.2), methods
Maintainer Michael Lawrence <michafla@gene.com>
SystemRequirements GNU make
Description Low-level interface to functionality in Qt for efficiently drawing
dynamic graphics and handling basic user input.
License GPL (>= 2)
LinkingTo qtbase
NeedsCompilation yes
Repository CRAN
Date/Publication 2015-11-11 16:34:36

R topics documented:

add-scenes-items .................................................. 2
antialiasReplace .................................................. 3
dashReplace ......................................................... 3
fontReplace .......................................................... 4
glyphExpansionReplace ........................................... 4
item-properties ....................................................... 5
layout-accessors ..................................................... 5
lineWidthReplace ................................................... 6
painting ............................................................... 7
Qanviz ................................................................. 8
qflipy ................................................................. 9
qglyphs ............................................................... 9
qlayer ................................................................. 10
qplotView ............................................................ 12
qscene ............................................................... 13
qupdate .............................................................. 14
These functions constitute a vectorized API for adding graphical primitive items into a scene. Each primitive is represented by a separate canvas item. This is a different paradigm from the layer-based painting, which has a more appropriate design for most interactive graphics. Most of the time, the user should ignore this API.

Usage

```r
qscene.points(s, x, y, radius=1)
qscene.lines(s, x, y, lwd=0)
qscene.segments(s, x1, y1, x2, y2, lwd=0)
qscene.rect(s, x, y, w=1, h=1)
```

Arguments

- `s` - The scene, a C++ QGraphicsScene
- `x` - The X coordinates, as expected by `xy.coords`, recycled
- `y` - The Y coordinates, as expected by `xy.coords`, recycled
- `radius` - The scalar radius for the points, in device coordinates
- `lwd` - The vector of line widths, in device coordinates, recycled
- `x1` - The first X coordinates, recycled
- `y1` - The first Y coordinates, recycled
- `x2` - The second X coordinates, recycled
- `y2` - The second Y coordinates, recycled
- `w` - Vector of rectangle widths, recycled
- `h` - Vector of rectangle heights, recycled

Author(s)

Deepayan Sarkar
antialiasReplace  Antialiasing

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enables or disables antialiasing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>qantialias(x) &lt;- value</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
</tr>
<tr>
<td>value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michael Lawrence</td>
</tr>
</tbody>
</table>

dashReplace  Dash patterns

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets the dash pattern</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>qdash(x) &lt;- value</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
</tr>
<tr>
<td>value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michael Lawrence</td>
</tr>
</tbody>
</table>
**fontReplace**  
*Fonts*

**Description**
Sets the font.

**Usage**
```r
qfont(x) <- value
```

**Arguments**
- `x`  
The paint context
- `value`  
The font, as a C++ `QFont` object

**Author(s)**
Michael Lawrence

---

**glyphExpansionReplace**  
*Dash patterns*

**Description**
Sets the glyph expansion, equivalent to `cex` in base R

**Usage**
```r
qglyphExpansion(x) <- value
```

**Arguments**
- `x`  
The paint context
- `value`  
Floating point multiplier of the glyph size

**Author(s)**
Michael Lawrence
item-properties  

**Item properties**

**Description**

Set properties of all items in a scene. Fast path for when a scene contains many items.

**Usage**

```cpp
qsetItemFlags(x, flag=Qt$QGraphicsItem$ItemIsMovable, status=FALSE)
qsetTextItemInteraction(x, mode=c("none", "editor", "browser"))
```

**Arguments**

- **x**
  - The scene, a C++ QGraphicsScene
- **flag**
  - A value or combination of values from the QGraphicsItem::GraphicsItemFlag enumeration
- **status**
  - Whether the flag should be set to TRUE or FALSE
- **mode**
  - Whether a text item (C++ QGraphicsTextItem) should behave as a text editor, text browser, or not allow interaction.

**Author(s)**

Deepayan Sarkar and Michael Lawrence

layout-accessors  

**Grid layout accessors**

**Description**

Add or retrieve a layer to or from the grid layout of another layer

**Usage**

```r
## S3 replacement method for class `Qanviz::Layer`
x[i=0, j=0, rowSpan=1, colSpan=1] <- value
## S3 method for class `Qanviz::Layer`
x[i=0, j=0]
```
Arguments

- **x**: Parent layer
- **i**: 0-based row position
- **j**: 0-based column position
- **rowSpan**: Number of rows spanned by `value`
- **colSpan**: Number of columns spanned by `value`
- **value**: The layer to add to the layout

Author(s)

Michael Lawrence

---

**lineWidthReplace**  
*Line width*

Description

Sets the line width

Usage

```r
qLineWidth(x) <- value
```

Arguments

- **x**: The paint context
- **value**: A numeric value indicating the line width, in device coords

Author(s)

Michael Lawrence
Description

These functions constitute the primary drawing API. There is support for drawing points, polylines, segments, circles, rectangles, polygons, vector paths, text, images and plot glyphs.

Usage

\begin{verbatim}
qdrawLine(p, x, y, stroke)
qdrawSegment(p, x0, y0, x1, y1, stroke)
qdrawPoint(p, x, y, stroke)
qdrawRect(p, xleft, ybottom, xright, ytop, stroke, fill)
qdrawCircle(p, x, y, r, stroke, fill)
qdrawPolygon(p, x, y, stroke, fill)
qdrawPath(p, path, stroke, fill)
qdrawText(p, text, x, y, halign=c("center", "left", "right"), valign=c("center", "basecenter", "baseline", "bottom", "top"), rot=0, color, cex=1, hceX=cex, vceX=cex)
qdrawImage(p, image, x, y)
qdrawGlyph(p, path, x, y, cex, stroke, fill)
\end{verbatim}

Arguments

\begin{itemize}
\item \texttt{p} The paint context
\item \texttt{x} The X coordinate vector, recycled. For polygons and polylines, NA values separate the graphical primitives.
\item \texttt{y} The Y coordinate vector, recycled. For polygons and polylines, NA values separate the graphical primitives.
\item \texttt{stroke} The vector of stroke colors, either a C++ \texttt{QColor} object, a matrix returned by \texttt{col2rgb} or any valid input to \texttt{col2rgb}, recycled
\item \texttt{x0} The vector of first X coordinates, recycled
\item \texttt{y0} The vector of first Y coordinates, recycled
\item \texttt{x1} The vector of second X coordinates, recycled
\item \texttt{y1} The vector of second Y coordinates, recycled
\item \texttt{fill} The vector of fill colors, either a C++ \texttt{QColor} object, a matrix returned by \texttt{col2rgb} or any valid input to \texttt{col2rgb}, recycled
\item \texttt{xleft} The vector of left X coordinates for a rectangle, recycled
\item \texttt{ybottom} The vector of bottom Y coordinates for a rectangle, recycled
\item \texttt{xright} The vector of right X coordinates for a rectangle, recycled
\item \texttt{ytop} The vector of top Y coordinates for a rectangle, recycled
\item \texttt{r} The radius of the circle, in device coordinates, recycled
\end{itemize}
**path**
A C++ QPainterPath object describing the glyph, or a list of such objects for qdrawPath.

**text**
The vector of strings to draw, recycled

**halign**
The side of the text to horizontally align to the coordinate

**valign**
The side of the text to vertically align to the coordinate. Besides the obvious alignment options, there are two different ways to center the text: according to the entire text extents ("center") or only according to the region above the baseline ("basecenter"). The former may be better for plotting text, while the latter may be better for labeling.

**rot**
The vector of the text rotations, in degrees, recycled

**color**
The stroke color of the text

**cex**
The vector of numeric expansion factors of the glyphs, recycled

**hcex**
The vector of numeric horizontal expansion factors of the glyphs, recycled. Overrides the cex in the horizontal direction.

**vcex**
The vector of numeric vertical expansion factors of the glyphs, recycled. Overrides the cex in the vertical direction.

**image**
A C++ QImage object

---

**Qanviz smoke library**

**Description**

The smoke library object representing the Qanviz library underlying qtpaint. It is not usually necessary to manipulate this object directly.

**Author(s)**

Michael Lawrence
**qflipy**

Flip the Y axis

---

**Description**

Generate transform for flipping Y axis.

**Usage**

```r
qflipY(ymax, ymin=0)
```

---

**Arguments**

- `ymax`: Maximum Y value or a rectangle (QRect or matrix)
- `ymin`: Minimum Y value

**Value**

`qflipY`: A QTransform object that will transform points by flipping the axis.

**Author(s)**

Michael Lawrence

**See Also**

- `qvmap`

---

**qglyphs**

Glyph constructors

---

**Description**

Some glyph constructors for simple glyphs like circle, square and triangle. For use with `qdrawGlyph`.

**Usage**

```r
qglyphCircle(r=5)
qglyphSquare(x=5)
qglyphTriangle(x=5, direction=c("up", "down"))
qglyphText(text="X", size=12)
qglyphSegment(x=5, direction=0)
```
Arguments

- **r**  
  Radius of the circle

- **x**  
  Length of one side of the square or triangle or the segment

- **direction**  
  Whether the triangle is pointing up or down, or the direction of the segment (unit: radian)

- **text**  
  The text of the text glyph

- **size**  
  The font size of the text glyph

Value

- `qglyphCircle`: C++ QPainterPath instance for use with `qdrawGlyph`

Author(s)

- Michael Lawrence

Description

This function constructs a Layer, an item in the canvas that may paint part of the plot and/or respond to user input. The behavior of the layer is implemented using R functions, which are passed to the constructor. Other arguments determine the scaling and positioning of the layer, and whether the drawing of the layer is cached and/or clipped.

Usage

```r
qlayer(parent=NULL, paintFun=NULL, keyPressFun=NULL, keyReleaseFun=NULL, mouseDoubleClickFun=NULL, mouseMoveFun=NULL, mousePressFun=NULL, mouseReleaseFun=NULL, wheelFun=NULL, hoverMoveFun=NULL, hoverEnterFun=NULL, hoverLeaveFun=NULL, contextMenuFun=NULL, dragEnterFun=NULL, dragLeaveFun=NULL, dragMoveFun=NULL, dropFun=NULL, focusInFun=NULL, focusOutFun=NULL, sizeHintFun=NULL, limits=qrect(), row=0, col=0, rowSpan=1, colSpan=1, geometry=defaultLayerGeometry(parent), clip=cache, cache=FALSE)
```
**Arguments**

- **parent**
  - The scene, for a top-level layer, or the parent layer that contains the new layer in a grid layout.

- **paintFun**
  - The function that implements painting, called whenever the layer needs to be repainted. All drawing occurs in data/layer coordinates.

- **keyPressFun**
  - The function called when a key is pressed.

- **keyReleaseFun**
  - The function called when a key is released.

- **mouseDoubleClickFun**
  - The function called when a mouse button is double-clicked.

- **mouseMoveFun**
  - The function called when the mouse is moved while holding down a button.

- **mousePressFun**
  - The function called when a mouse button is pressed.

- **mouseReleaseFun**
  - The function called when a mouse button is released.

- **wheelFun**
  - The function called when the mouse wheel is turned.

- **hoverMoveFun**
  - The function called when the mouse moves without any buttons pressed.

- **hoverEnterFun**
  - The function called when the mouse pointer enters the layer.

- **hoverLeaveFun**
  - The function called when the mouse pointer leaves the layer.

- **contextMenuFun**
  - The function called when a context menu is requested, such as through a right mouse button click.

- **dragEnterFun**
  - The function called when the pointer enters the layer while dragging something.

- **dragLeaveFun**
  - The function called when the pointer leaves the layer while dragging something.

- **dragMoveFun**
  - The function called when the pointer moves within the layer while dragging something.

- **dropFun**
  - The function called when something is dropped on the layer.

- **focusInFun**
  - The function called when the layer gains the keyboard focus.

- **focusOutFun**
  - The function called when the layer loses the keyboard focus.

- **sizeHintFun**
  - The function called to determine the size constraints of the layer. Not currently documented.

- **limits**
  - A QRectF, possibly created by qrect, indicating the X and Y scales of the layer in data/layer coordinates.

- **row**
  - The 0-based row index of the layer in the parent grid layout.

- **col**
  - The 0-based column index of the layer in the parent grid layout.

- **rowSpan**
  - The 0-based number of rows spanned by the layer in the layout.

- **colSpan**
  - The 0-based number of cols spanned by the layer in the layout.

- **geometry**
  - A QRectF, possibly created by qrect, indicating the position and size of the layer in figure/scene coordinates. This is overridden by the parent grid layout, so is really only useful for a top-level layer. A warning will be issued if the geometry is specified along with a parent layer. We also issue a warning if this argument is specified when the scene has a view in “geometry” rescale mode, because the view determines the geometry. The default geometry is the bounding rectangle of the scene, if not null, or 600x400 otherwise.
### Details

All drawing and user input handling is performed by R callbacks, which must accept a specific set of arguments. The paint callback, passed as `paintFun`, must take at least two arguments, conventionally named `layer` and `painter`. The `layer` argument is a C++ `RLayer` object, the same instance that was created by calling the constructor. All painting is performed through the `painter` argument, which is a C++ `Painter` object. See the `paint functions` for more details. The `paintFun` may take one additional, optional argument, conventionally named `exposed`, which is the rectangle, in layer coordinates, that needs to be drawn.

All of the other callbacks, except for `sizeHintFun`, are event handlers. Two arguments are passed, conventionally named `layer` and `event`. The former is the layer constructed in the call to `qlayer`, and the latter describes the event as an instance of C++ `QGraphicsSceneEvent`. The exact subclass depends on the event. Manipulating an event currently requires low-level calls through the `qtbase` package. See its documentation.

### Value

The layer, a C++ instance of `RLayer`

### Author(s)

Michael Lawrence

### Examples

```r
scene <- qscene()
layer <- qlayer(scene, function(layer, painter) {
  qdrawCircle(painter, 1:10, 1:10, 1)
}, limits = qtbase::qrect(0, 0, 11, 11))
qplotView(scene)
```

### Description

An extension of `QGraphicsView` with special functionality for plotting statistical graphics.

### Usage

```r
qplotView(scene, parent, rescale=c("geometry", "transform", "none"), opengl=TRUE)
```
**Arguments**

- `scene`: The scene, a QGraphicsScene
- `parent`: The parent QObject, usually a QWidget to contain the view
- `rescale`: The rescale mode, see details
- `opengl`: If TRUE, use OpenGL, otherwise the software driver

**Details**

The PlotView class extends QGraphicsView to add two new features, from the user perspective. First, it automates rescaling when the widget is resized. There are three rescale modes: geometry, transform and none:

- **geometry**: Most similar to how base R graphics devices behave: the geometry of the figure is fixed to the geometry of the view. This mode is often convenient but really only works if there is only a single view for each scene.
- **transform**: Adjusts the view transform so that the visible region of the scene does not change due to size changes. This is most similar to the behavior of GGobi and supports multiple views of the same scene. The downside is that the layout is not activated, so it cannot adapt to make better use of the available space. It works best with graphics that overlay guides on the plot, rather than position them adjacent to the plot in a layout.
- **none**: No rescaling is performed; when the size changes, more or less of the plot is shown. This is probably the most common mode in zoomable user interfaces.

The other feature is the overlay scene: a separate scene that is fixed to the geometry of the viewport. It is always shown over the primary scene and it is stationary across transformations and scrolling of the viewport. This is useful for overlaying guides on a plot in a fixed position, like the axes in GGobi. Call the `overlay` method on a plot view instance to obtain the overlay scene and manipulate it directly.

**Value**

A C++ PlotView object

**Author(s)**

Michael Lawrence

---

**Description**

Create a scene, a container of layers

**Usage**

```cpp
qscene(parent)
```
Arguments

parent  If non-NULL, a C++ QObject object that owns the memory of the scene

Details

Often, a PlotView object is passed as parent, so that the scene is not deleted until the view is destroyed.

Value

A C++ QGraphicsScene object

Author(s)

Michael Lawrence

update  Updating drawings

Description

Force a redraw of a layer or scene, clearing the cache. This needs to be called whenever the drawing would change, e.g., if the data or some visual attribute has changed. There is no automatic way for qtpaint to detect this.

Usage

update(x)
## S3 method for class 'QGraphicsView'
update(x)
## S3 method for class 'QGraphicsScene'
update(x)
## S3 method for class 'QGraphicsItem'
update(x)

Arguments

x  The object, usually a layer or scene, to be redrawn

Author(s)

Michael Lawrence
**qvmmap**

*Mapping coordinates*

**Description**

Transforms X and Y coordinates with a Qt-style transformation matrix. The advantage over direct use of Qt is vectorization.

**Usage**

```r
qvmmap(m, x, y)
```

**Arguments**

- `m` A matrix encoding the transformation, or something coercible to a matrix, like a C++ QTransform instance
- `x` X coordinates; if `y` is missing, should be something coercible to a numeric vector or matrix. If the vector coercion succeeds, the vector is coerced to a matrix with `matrix(x, ncol = 2, byrow = TRUE)`. The first column is taken as X, the second as Y.
- `y` Y coordinates, optional

**Value**

The mapped coordinates, as a two column (X, Y) matrix, unless `y` is missing, in which case an attempt is made to coerce the result to the class of `x`, if any.

**Author(s)**

Michael Lawrence

---

**stroke-fill**

*Stroke and fill colors*

**Description**

Functions for controlling the stroke and fill colors, including the ability to disable or enable stroking and filling.

**Usage**

```r
qhasStroke(x) <- value
qhasFill(x) <- value
qstrokeColor(x) <- value
qfillColor(x) <- value
```
Arguments

- \textit{x}: The paint context
- \textit{value}: The stroke or fill color, or a logical value for \texttt{qhasStroke} and \texttt{qHasFill}.

Value

\texttt{qhasStroke}: The stroke or fill color, or a logical value for \texttt{qHasStroke} and \texttt{qHasFill}.

Author(s)

- Michael Lawrence

---

text-extents \hspace{1cm} \textit{Text extents}

Description

Get text extents and font metrics

Usage

- \texttt{qtextExtents(p, text)}
- \texttt{qstrWidth(p, text)}
- \texttt{qstrHeight(p, text)}
- \texttt{qfontMetrics(p)}

Arguments

- \textit{p}: The paint context
- \textit{text}: The text to analyze

Value

- \texttt{qtextExtents}: A matrix representing the text bounds for \texttt{qtextExtents}, a number for \texttt{qstrWidth} and \texttt{qstrHeight}, or a vector with the ascent and descent for \texttt{qfontMetrics}

Author(s)

- Michael Lawrence
transform

User to device transform

Description
Access the user to device coordinate transform of the paint context.

Usage
qdeviceTransform(x)
qdeviceTransform(x) <- value
qdeviceTransformEnabled(x) <- value

Arguments
x The paint context
value The desired user to device transform, as a C++ QTransform object, or a logical scalar for qdeviceTransformEnabled.

Value
qdeviceTransform: The user to device transform, as a C++ QTransform object

Author(s)
Michael Lawrence
## Index

<table>
<thead>
<tr>
<th>.Qanviz::Layer (layout-accessors)</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>[&lt;- .Qanviz::Layer (layout-accessors)</td>
<td>5</td>
</tr>
<tr>
<td>add-scene-items</td>
<td>2</td>
</tr>
<tr>
<td>antialiasReplace</td>
<td>3</td>
</tr>
<tr>
<td>col2rgb</td>
<td>7</td>
</tr>
<tr>
<td>dashReplace</td>
<td>3</td>
</tr>
<tr>
<td>fontReplace</td>
<td>4</td>
</tr>
<tr>
<td>glyphExpansionReplace</td>
<td>4</td>
</tr>
<tr>
<td>item-properties</td>
<td>5</td>
</tr>
<tr>
<td>layout-accessors</td>
<td>5</td>
</tr>
<tr>
<td>lineWidthReplace</td>
<td>6</td>
</tr>
<tr>
<td>painting</td>
<td>7</td>
</tr>
<tr>
<td>quantialias&lt;- (antialiasReplace)</td>
<td>3</td>
</tr>
<tr>
<td>Qanviz</td>
<td>8</td>
</tr>
<tr>
<td>qdash&lt;- (dashReplace)</td>
<td>3</td>
</tr>
<tr>
<td>qdeviceTransform (transform)</td>
<td>17</td>
</tr>
<tr>
<td>qdeviceTransform&lt;- (transform)</td>
<td>17</td>
</tr>
<tr>
<td>qdeviceTransformEnabled&lt;- (transform)</td>
<td>17</td>
</tr>
<tr>
<td>qdrawCircle (painting)</td>
<td>7</td>
</tr>
<tr>
<td>qdrawGlyph</td>
<td>9</td>
</tr>
<tr>
<td>qdrawGlyph (painting)</td>
<td>7</td>
</tr>
<tr>
<td>qdrawImage (painting)</td>
<td>7</td>
</tr>
<tr>
<td>qdrawLine (painting)</td>
<td>7</td>
</tr>
<tr>
<td>qdrawPath (painting)</td>
<td>7</td>
</tr>
<tr>
<td>qdrawPoint (painting)</td>
<td>7</td>
</tr>
<tr>
<td>qdrawPolygon (painting)</td>
<td>7</td>
</tr>
<tr>
<td>qdrawRect (painting)</td>
<td>7</td>
</tr>
<tr>
<td>qdrawSegment (painting)</td>
<td>7</td>
</tr>
<tr>
<td>qdrawText (painting)</td>
<td>7</td>
</tr>
<tr>
<td>qfillColor&lt;- (stroke-fill)</td>
<td>15</td>
</tr>
<tr>
<td>qflipY (qflipy)</td>
<td>9</td>
</tr>
<tr>
<td>qflipy</td>
<td>9</td>
</tr>
<tr>
<td>qfont&lt;- (fontReplace)</td>
<td>4</td>
</tr>
<tr>
<td>qfontMetrics (text-extents)</td>
<td>16</td>
</tr>
<tr>
<td>qglyphCircle (qglyphs)</td>
<td>9</td>
</tr>
<tr>
<td>qglyphExpansion&lt;- (glyphExpansionReplace)</td>
<td>4</td>
</tr>
<tr>
<td>qglyphs</td>
<td>9</td>
</tr>
<tr>
<td>qglyphSegment (qglyphs)</td>
<td>9</td>
</tr>
<tr>
<td>qglyphSquare (qglyphs)</td>
<td>9</td>
</tr>
<tr>
<td>qglyphText (qglyphs)</td>
<td>9</td>
</tr>
<tr>
<td>qglyphTriangle (qglyphs)</td>
<td>9</td>
</tr>
<tr>
<td>qhasFill&lt;- (stroke-fill)</td>
<td>15</td>
</tr>
<tr>
<td>qhasStroke&lt;- (stroke-fill)</td>
<td>15</td>
</tr>
<tr>
<td>qlayer</td>
<td>10</td>
</tr>
<tr>
<td>qlineWidth&lt;- (lineWidthReplace)</td>
<td>6</td>
</tr>
<tr>
<td>qplotView</td>
<td>12</td>
</tr>
<tr>
<td>qrect</td>
<td>11</td>
</tr>
<tr>
<td>qscene</td>
<td>13</td>
</tr>
<tr>
<td>qscene.lines (add-scene-items)</td>
<td>2</td>
</tr>
<tr>
<td>qscene.points (add-scene-items)</td>
<td>2</td>
</tr>
<tr>
<td>qscene.rect (add-scene-items)</td>
<td>2</td>
</tr>
<tr>
<td>qscene.segments (add-scene-items)</td>
<td>2</td>
</tr>
<tr>
<td>qsetItemFlags (item-properties)</td>
<td>5</td>
</tr>
<tr>
<td>qsetTextItemInteraction (item-properties)</td>
<td>5</td>
</tr>
<tr>
<td>qstrHeight (text-extents)</td>
<td>16</td>
</tr>
<tr>
<td>qstrokeColor&lt;- (stroke-fill)</td>
<td>15</td>
</tr>
<tr>
<td>qstrWidth (text-extents)</td>
<td>16</td>
</tr>
<tr>
<td>qtextExtents (text-extents)</td>
<td>16</td>
</tr>
<tr>
<td>qupdate</td>
<td>14</td>
</tr>
<tr>
<td>qvmap</td>
<td>9</td>
</tr>
<tr>
<td>stroke-fill</td>
<td>15</td>
</tr>
<tr>
<td>text-extents</td>
<td>16</td>
</tr>
<tr>
<td>transform</td>
<td>17</td>
</tr>
<tr>
<td>xy.coords</td>
<td>2</td>
</tr>
</tbody>
</table>