QT Methods and Functions

Nathan Warner

Computer Science
Northern Illinois University
December 12, 2023
United States

Contents

Known Includes

- include <QApplication> // Manages application-wide resources and the main event loop
- include <QString> // String class for handling Unicode text
- include <QWidget> // Base class for all UI objects in Qt
- include <QPushButton> // Provides a push button widget
- include <QLabel> // Provides a text or image display widget
- include <QProcess> // Enables running external processes
- include <QStringList> // List of QString objects, often used for string manipulation
- include <QPainter> // Used for drawing graphics in widgets
- include <QPoint> // Represents x and y coordinates in a 2D space
- include <QtCore> // For all shape objects
- include <QRect> // Defines a rectangle in the plane using integer precision
- include <QPolygon> // Represents a polygon defined by a vector of points
- include <QBrush> // Used for filling shapes with solid colors, patterns, or gradients
- include <QPen> // Used for drawing lines and outlines of shapes
- include <QImage> // Represents an image; used in conjunction with QPainter
- include <QColor> // Used to define html colors
- include <QGradient> // To create gradient objects
- include <QLayout> // Used for layouts
- include <QHBoxLayout> // Used for layouts
- include <QVBoxLayout> // Used for layouts
- include <QFont> // Used for fonts and font styles

Known Objects

- **QString**
- **QStringList**
- **QPushButton**
- **QLabel**
- **QFont**
- **QColor**
- **QLinearGradient**
- **QRadialGradient**
- **QConicalGradient**
- **Qpen**
- **QBrush**
- **Colors**
- **QPainter**
- **Shape Objects**
- **QHBoxLayout**
- **QVBoxLayout**
- **QGridLayout**

Constructors

- **QString**:
  - `QString()`
  - `QString(const QString &)`
  - `QString(const char *)`
- **QStringList**:
  - `QStringList()`
  - `QStringList(const QString &)`
- **QPushButton**:
  - `QPushButton(QWidget *parent = nullptr)`
  - `QPushButton(const QString &text, QWidget *parent = nullptr)`
- **QLabel**:
  - `QLabel(QWidget *parent = nullptr, Qt::WindowFlags f = Qt::WindowFlags())`
  - `QLabel(const QString &text, QWidget *parent = nullptr, Qt::WindowFlags f = Qt::WindowFlags())`
- **QFont**:
  - `QFont()`
  - `QFont(const QString &family, int pointSize = -1, int weight = -1, bool italic = false)`
- **QColor**:
  - `QColor()`
  - `QColor(int r, int g, int b, int a = 255)`
  - `QColor(const QString &name)`
  - `QColor(Qt::GlobalColor color)`
- **QLinearGradient**:
  - `QLinearGradient()`
  - `QLinearGradient(const QPointF &start, const QPointF &finalStop)`
  - `QLinearGradient(qreal xStart, qreal yStart, qreal xFinalStop, qreal yFinalStop)`
- **QRadialGradient**:
  - `QRadialGradient()`
  - `QRadialGradient(const QPointF &center, qreal radius, const QPointF &focalPoint)`
  - `QRadialGradient(qreal cx, qreal cy, qreal radius, qreal fx, qreal fy)`
- **QConicalGradient**:
  - `QConicalGradient()`
  - `QConicalGradient(const QPointF &center, qreal startAngle)`
  - `QConicalGradient(qreal cx, qreal cy, qreal startAngle)`

- **QPen**:
  - `QPen()`
  - `QPen(const QColor &color)`
  - `QPen(Qt::PenStyle style)`
  - `QPen(const QBrush &brush, qreal width, Qt::PenStyle style = Qt::SolidLine, Qt::PenCapStyle cap = Qt::SquareCap, Qt::PenJoinStyle join = Qt::BevelJoin)`
- **QBrush**:
  - `QBrush()`
  - `QBrush(Qt::BrushStyle style)`
  - `QBrush(const QColor &color, Qt::BrushStyle style = Qt::SolidPattern)`
  - `QBrush(const QPixmap &pixmap)`
  - `QBrush(const QBrush &brush)`
- **QPainter**: (No default constructor; use `begin()` and `end()` methods)
- **Shape Objects** (e.g., QRect, QPoint):
  - `QRect()`
  - `QRect(int x, int y, int width, int height)`
  - `QPoint()`
  - `QPoint(int xpos, int ypos)`
- **QHBoxLayout**:
  - `QHBoxLayout()`
  - `QHBoxLayout(QWidget *parent)`
- **QVBoxLayout**:
  - `QVBoxLayout()`
  - `QVBoxLayout(QWidget *parent)`
- **QGridLayout**:
  - `QGridLayout()`
  - `QGridLayout(QWidget *parent)`

QWidget Methods

- **Constructor** $\mapsto$ `QWidget(QWidget *parent = nullptr, Qt::WindowFlags f = Qt::WindowFlags())`: Initializes a new instance of QWidget with optional parent and window flags.
- **setGeometry** $\mapsto$ `void`: Sets the geometry of the widget.
- **geometry** $\mapsto$ `QRect`: Returns the widget's geometry.
- **move** $\mapsto$ `void`: Moves the widget to a specified position.
- **resize** $\mapsto$ `void`: Resizes the widget.
- **setFixedSize** $\mapsto$ `void`: Sets a fixed size for the widget.
- **setStyle** $\mapsto$ `void`: Sets the style of the widget.
- **setStyleSheet** $\mapsto$ `void`: Sets the style sheet used for custom widget styling.
- **update** $\mapsto$ `void`: Updates the widget.
- **repaint** $\mapsto$ `void`: Repaints the widget immediately.
- **show** $\mapsto$ `void`: Shows the widget.
- **hide** $\mapsto$ `void`: Hides the widget.
- **setVisible** $\mapsto$ `void`: Sets the visibility of the widget.
- **close** $\mapsto$ `bool`: Closes the widget.
- **setLayout** $\mapsto$ `void`: Sets the layout for the widget.
- **setFocus** $\mapsto$ `void`: Sets focus to the widget.
- **clearFocus** $\mapsto$ `void`: Clears focus from the widget.
- **setFocusPolicy** $\mapsto$ `void`: Sets the focus policy for the widget.
- **windowTitle** / **setWindowTitle** $\mapsto$ `QString` / `void`: Gets or sets the window title.
- **setProperty** $\mapsto$ `void`: Sets a property of the widget.
- **property** $\mapsto$ `QVariant`: Returns the value of a property.

QString Methods

- **length()** const ↦ `int`: Returns the length of the string.
- **isEmpty()** const ↦ `bool`: Returns true if the string is empty.
- **append(const QString &str)** ↦ `QString &`: Appends the given string to this string.
- **prepend(const QString &str)** ↦ `QString &`: Prepends the given string to this string.
- **contains(const QString &str, Qt::CaseSensitivity cs = Qt::CaseSensitive)** const ↦ `bool`: Returns true if the string contains the given substring.
- **indexOf(const QString &str, int from = 0, Qt::CaseSensitivity cs = Qt::CaseSensitive)** const ↦ `int`: Returns the index position of the first occurrence of the given substring.
- **lastIndexOf(const QString &str, int from = -1, Qt::CaseSensitivity cs = Qt::CaseSensitive)** const ↦ `int`: Returns the index position of the last occurrence of the given substring.
- **remove(int pos, int len)** ↦ `QString &`: Removes `len` characters from the string starting at position `pos`.
- **replace(const QString &before, const QString &after, Qt::CaseSensitivity cs = Qt::CaseSensitive)** ↦ `QString &`: Replaces occurrences of the substring `before` with the substring `after`.
- **split(const QString &delimiter, Qt::SplitBehavior splitBehavior = Qt::KeepEmptyParts, Qt::CaseSensitivity cs = Qt::CaseSensitive)** const ↦ `QStringList`: Splits the string into a list of strings divided by the given delimiter.
- **toLower()** const ↦ `QString`: Returns a copy of the string converted to lowercase.
- **toUpper()** const ↦ `QString`: Returns a copy of the string converted to uppercase.
- **trimmed()** const ↦ `QString`: Returns a copy of the string with whitespace removed from the start and end.
- **left(int n)** const ↦ `QString`: Returns the leftmost `n` characters of the string.
- **right(int n)** const ↦ `QString`: Returns the rightmost `n` characters of the string.
- **mid(int position, int n = -1)** const ↦ `QString`: Returns a substring of `n` characters from the string starting at `position`.

QStringList Methods

- **append(const QString &str)**: Adds the given string to the end of the list.
- **at(int i) const** $\mapsto$ `QString`: Returns the string at the specified position in the list.
- **join(const QString &separator) const** $\mapsto$ `QString`: Concatenates all the strings in the list into a single string with a specified separator.
- **sort(Qt::SortOrder order = Qt::AscendingOrder)**: Sorts the list in ascending or descending order.
- **filter(const QString &pattern, Qt::CaseSensitivity cs = Qt::CaseSensitive) const** $\mapsto$ `QStringList`: Returns a new list containing only the strings that match a given pattern.
- **size() const / count() const** $\mapsto$ `int`: Returns the number of items in the list.
- **isEmpty() const** $\mapsto$ `bool`: Checks if the list is empty.
- **clear()**: Clears all items from the list.
- **removeDuplicates()** $\mapsto$ `int`: Removes duplicate strings from the list and returns the number of removed items.
- **contains(const QString &str, Qt::CaseSensitivity cs = Qt::CaseSensitive) const** $\mapsto$ `bool`: Returns true if the list contains the given string.
- **indexOf(const QString &str, int from = 0) const** $\mapsto$ `int`: Returns the index of the first occurrence of the string in the list, searching forward from index `from`.
- **replaceInStrings(const QString &before, const QString &after, Qt::CaseSensitivity cs = Qt::CaseSensitive)**: Replaces occurrences of a substring within all the strings of the list.

Button Methods

- **setText(const QString &text)** $\mapsto$ `void`: Sets the button's text to `text`.
- **text()** const $\mapsto$ `QString`: Returns the button's text.
- **setIcon(const QIcon &icon)** $\mapsto$ `void`: Sets the icon of the button to `icon`.
- **icon()** const $\mapsto$ `QIcon`: Returns the button's icon.
- **setShortcut(const QKeySequence &key)** $\mapsto$ `void`: Sets a shortcut key for the button with `key`.
- **shortcut()** const $\mapsto$ `QKeySequence`: Returns the shortcut key associated with the button.
- **setChecked(bool check)** $\mapsto$ `void`: Sets the check state of the button to `check` (if the button is checkable).
- **isChecked()** const $\mapsto$ `bool`: Returns true if the button is checked.
- **setFlat(bool flat)** $\mapsto$ `void`: Sets whether the button is flat to `flat`.
- **isFlat()** const $\mapsto$ `bool`: Returns true if the button is flat.
- **setMenu(QMenu *menu)** $\mapsto$ `void`: Sets the associated drop-down menu of the button to `menu`.
- **menu()** const $\mapsto$ `QMenu*`: Returns the associated menu of the button.
- **showMenu()** $\mapsto$ `void`: Displays the associated drop-down menu.
- **setAutoDefault(bool autoDefault)** $\mapsto$ `void`: Sets whether the button is an auto default button to `autoDefault`.
- **isAutoDefault()** const $\mapsto$ `bool`: Returns true if the button is an auto default button.
- **setDefault(bool default)** $\mapsto$ `void`: Sets whether the button is the default button to `default`.
- **isDefault()** const $\mapsto$ `bool`: Returns true if the button is the default button.
- **setCheckable(bool checkable)** $\mapsto$ `void`: Sets whether the button is checkable to `checkable`.
- **isCheckable()** const $\mapsto$ `bool`: Returns true if the button is checkable.
- **click()** $\mapsto$ `void`: Simulates a click on the button.
- **animateClick(int msec = 100)** $\mapsto$ `void`: Simulates an animated click on the button, with the animation lasting `msec` milliseconds.
- **setAutoRepeat(bool autoRepeat)** $\mapsto$ `void`: Enables or disables auto-repeat for the button to `autoRepeat`.
- **autoRepeat()** const $\mapsto$ `bool`: Returns true if auto-repeat is enabled for the button.
- **setAutoRepeatDelay(int delay)** $\mapsto$ `void`: Sets the auto-repeat delay for the button to `delay` milliseconds.
- **autoRepeatDelay()** const $\mapsto$ `int`: Returns the auto-repeat delay in milliseconds.

- **setAutoRepeatInterval(int interval)** $\mapsto$ `void`: Sets the auto-repeat interval for the button to `interval` milliseconds.
- **autoRepeatInterval()** const $\mapsto$ `int`: Returns the auto-repeat interval in milliseconds.

7.1 Signals

- **clicked(bool checked)**: Emitted when the button is clicked. If the button is checkable, `checked` is true if the button is checked, otherwise false.
- **pressed()**: Emitted when the button is pressed down.
- **released()**: Emitted when the button is released.
- **toggled(bool checked)**: Emitted when the toggle state of the button changes. `checked` is true if the button is checked, otherwise false.

Label Methods

- **setText(const QString &)** $\mapsto$ `void`: Sets the label's text.
- **setPixmap(const QPixmap &)** $\mapsto$ `void`: Sets the label's pixmap.
- **setAlignment(Qt::Alignment)** $\mapsto$ `void`: Sets the alignment of the label's content.
- **setWordWrap(bool)** $\mapsto$ `void`: Enables or disables word wrapping.
- **text() const** $\mapsto$ `QString`: Returns the label's text.
- **pixmap() const** $\mapsto$ `const QPixmap *`: Returns the label's pixmap.
- **alignment() const** $\mapsto$ `Qt::Alignment`: Returns the alignment of the label's content.
- **wordWrap() const** $\mapsto$ `bool`: Returns whether word wrapping is enabled.
- **clear()** $\mapsto$ `void`: Clears the label's contents.
- **setIndent(int)** $\mapsto$ `void`: Sets the indent used for the label's text or pixmap.
- **indent() const** $\mapsto$ `int`: Returns the indent used for the label's text or pixmap.
- **setMargin(int)** $\mapsto$ `void`: Sets the margin around the label's contents.
- **margin() const** $\mapsto$ `int`: Returns the margin around the label's contents.
- **setOpenExternalLinks(bool)** $\mapsto$ `void`: Sets whether the label should open external links.
- **openExternalLinks() const** $\mapsto$ `bool`: Returns whether the label opens external links.

8.0.1 Signals

- **linkActivated(const QString &)**: Emitted when a link in the text is activated.
- **linkHovered(const QString &)**: Emitted when the mouse hovers over a link in the text.

QFont methods

- **setFamily(QString &family)** $\mapsto$ `void`: Sets the font family to `family`.
- **family()** const $\mapsto$ `QString`: Returns the family name of the font.
- **setPointSize(int size)** $\mapsto$ `void`: Sets the font size in points to `size`.
- **pointSize()** const $\mapsto$ `int`: Returns the point size of the font.
- **setPixelSize(int size)** $\mapsto$ `void`: Sets the font size in pixels to `size`.
- **pixelSize()** const $\mapsto$ `int`: Returns the pixel size of the font.
- **setWeight(int weight)** $\mapsto$ `void`: Sets the weight of the font to `weight`.
- **weight()** const $\mapsto$ `int`: Returns the weight of the font.
- **setBold(bool bold)** $\mapsto$ `void`: Sets the font's bold property to `bold`.
- **bold()** const $\mapsto$ `bool`: Returns true if the font is bold.
- **setItalic(bool italic)** $\mapsto$ `void`: Sets the font's italic property to `italic`.
- **italic()** const $\mapsto$ `bool`: Returns true if the font is italic.
- **setUnderline(bool underline)** $\mapsto$ `void`: Sets the font's underline property to `underline`.
- **underline()** const $\mapsto$ `bool`: Returns true if the font is underlined.
- **setOverline(bool overline)** $\mapsto$ `void`: Sets the font's overline property to `overline`.
- **overline()** const $\mapsto$ `bool`: Returns true if the font has an overline.
- **setStrikeOut(bool strikeOut)** $\mapsto$ `void`: Sets the font's strikeout property to `strikeOut`.
- **strikeOut()** const $\mapsto$ `bool`: Returns true if the font is struck out.
- **setKerning(bool enable)** $\mapsto$ `void`: Enables or disables kerning based on `enable`.
- **kerning()** const $\mapsto$ `bool`: Returns true if kerning is enabled.
- **setStyle(QFont::Style style)** $\mapsto$ `void`: Sets the style of the font to `style`.
- **style()** const $\mapsto$ `QFont::Style`: Returns the style of the font.
- **setStyleHint(QFont::StyleHint hint, QFont::StyleStrategy strategy = QFont::PreferDefault)** $\mapsto$ `void`: Sets the style hint and strategy of the font to `hint` and `strategy`.
- **styleHint()** const $\mapsto$ `QFont::StyleHint`: Returns the style hint of the font.
- **setStretch(int factor)** $\mapsto$ `void`: Sets the stretch factor of the font to `factor`.
- **stretch()** const $\mapsto$ `int`: Returns the stretch factor of the font.
- **setLetterSpacing(QFont::SpacingType type, qreal spacing)** $\mapsto$ `void`: Sets the type and amount of letter spacing.
- **letterSpacing()** const $\mapsto$ `qreal`: Returns the amount of letter spacing.
- **setWordSpacing(qreal spacing)** $\mapsto$ `void`: Sets the amount of word spacing to `spacing`.
- **wordSpacing()** const $\mapsto$ `qreal`: Returns the amount of word spacing.

QColor Methods

- **setRgb(int r, int g, int b, int a = 255)**: Sets the color using RGBA values.
- **setRgbF(qreal r, qreal g, qreal b, qreal a = 1.0)**: Sets the color using RGBA values as floating point numbers.
- **setNamedColor(const QString &name)**: Sets the color using a color name.
- **setHsl(int h, int s, int l, int a = 255)**: Sets the color using HSL values.
- **setHslF(qreal h, qreal s, qreal l, qreal a = 1.0)**: Sets the color using HSL values as floating point numbers.
- **red() const** $\mapsto$ `int`: Returns the red component of the color.
- **green() const** $\mapsto$ `int`: Returns the green component of the color.
- **blue() const** $\mapsto$ `int`: Returns the blue component of the color.
- **alpha() const** $\mapsto$ `int`: Returns the alpha (transparency) component of the color.
- **hue() const** $\mapsto$ `int`: Returns the hue component of the color.
- **saturation() const** $\mapsto$ `int`: Returns the saturation component of the color.
- **lightness() const** $\mapsto$ `int`: Returns the lightness component of the color.
- **darker(int factor = 200) const** $\mapsto$ `QColor`: Returns a darker color.
- **lighter(int factor = 150) const** $\mapsto$ `QColor`: Returns a lighter color.
- **isValid() const** $\mapsto$ `bool`: Returns true if the color is valid.
- **name(QColor::NameFormat format = QColor::HexRgb) const** $\mapsto$ `QString`: Returns the name of the color.
- **toRgb() const** $\mapsto$ `QColor`: Converts the color to an RGB color.
- **toHsl() const** $\mapsto$ `QColor`: Converts the color to an HSL color.

QGradient Methods

11.1 QGradient Class Methods

- **setColorAt(qreal position, const QColor &color)**: Sets the color at the specified position in the gradient.
- **setSpread(QGradient::Spread spread)**: Sets the spread method for the gradient.
- **spread() const** $\mapsto$ `QGradient::Spread`: Returns the current spread method.
- **setCoordinateMode(QGradient::CoordinateMode mode)**: Sets the coordinate mode of the gradient.
- **coordinateMode() const** $\mapsto$ `QGradient::CoordinateMode`: Returns the coordinate mode.
- **stops() const** $\mapsto$ `QList<QPair<qreal, QColor>>`: Returns the gradient stops as a list of pairs of positions and colors.

11.2 QLinearGradient Class Methods

- **setStart(const QPointF &start)**: Sets the start point of the linear gradient.
- **setFinalStop(const QPointF &stop)**: Sets the final stop point of the linear gradient.
- **start() const** $\mapsto$ `QPointF`: Returns the start point.
- **finalStop() const** $\mapsto$ `QPointF`: Returns the final stop point.

11.3 QRadialGradient Class Methods

- **setCenter(const QPointF &center)**: Sets the center of the radial gradient.
- **setRadius(qreal radius)**: Sets the radius of the radial gradient.
- **setFocalPoint(const QPointF &focalPoint)**: Sets the focal point of the radial gradient.
- **center() const** $\mapsto$ `QPointF`: Returns the center point.
- **radius() const** $\mapsto$ `qreal`: Returns the radius.
- **focalPoint() const** $\mapsto$ `QPointF`: Returns the focal point.

11.4 QConicalGradient Class Methods

- **setCenter(const QPointF &center)**: Sets the center of the conical gradient.
- **setAngle(qreal angle)**: Sets the start angle of the conical gradient.
- **center() const** $\mapsto$ `QPointF`: Returns the center point.
- **angle() const** $\mapsto$ `qreal`: Returns the start angle.

QPen Methods

- **setColor(const QColor &color)** $\mapsto$ `void`: Sets the color of the pen to `color`.
- **color()** const $\mapsto$ `QColor`: Returns the color of the pen.
- **setWidth(int width)** $\mapsto$ `void`: Sets the width of the pen to `width`.
- **width()** const $\mapsto$ `int`: Returns the width of the pen.
- **setBrush(const QBrush &brush)** $\mapsto$ `void`: Sets the brush of the pen to `brush`.
- **brush()** const $\mapsto$ `QBrush`: Returns the brush of the pen.
- **setStyle(Qt::PenStyle style)** $\mapsto$ `void`: Sets the style of the pen to `style`.
- **style()** const $\mapsto$ `Qt::PenStyle`: Returns the style of the pen.
- **setCapStyle(Qt::PenCapStyle capStyle)** $\mapsto$ `void`: Sets the cap style of the pen to `capStyle`.
- **capStyle()** const $\mapsto$ `Qt::PenCapStyle`: Returns the cap style of the pen.
- **setJoinStyle(Qt::PenJoinStyle joinStyle)** $\mapsto$ `void`: Sets the join style of the pen to `joinStyle`.
- **joinStyle()** const $\mapsto$ `Qt::PenJoinStyle`: Returns the join style of the pen.

QBrush Methods

- **setColor(const QColor &color)** $\mapsto$ `void`: Sets the color of the brush to `color`.
- **color()** const $\mapsto$ `QColor`: Returns the color of the brush.
- **setStyle(Qt::BrushStyle style)** $\mapsto$ `void`: Sets the style of the brush to `style`.
- **style()** const $\mapsto$ `Qt::BrushStyle`: Returns the style of the brush.
- **setTexture(const QPixmap &pixmap)** $\mapsto$ `void`: Sets the texture of the brush to the pixmap `pixmap`.
- **texture()** const $\mapsto$ `QPixmap`: Returns the pixmap used as the texture of the brush.
- **setTextureImage(const QImage &image)** $\mapsto$ `void`: Sets the texture of the brush to the image `image`.
- **textureImage()** const $\mapsto$ `QImage`: Returns the image used as the texture of the brush.
- **setMatrix(const QMatrix &matrix)** $\mapsto$ `void`: Sets the transformation matrix of the brush to `matrix`.
- **matrix()** const $\mapsto$ `QMatrix`: Returns the transformation matrix of the brush.

Colors for pen and brush

- **Qt::black**: Represents the color black.
- **Qt::white**: Represents the color white.
- **Qt::red**: Represents the color red.
- **Qt::green**: Represents the color green.
- **Qt::blue**: Represents the color blue.
- **Qt::cyan**: Represents the color cyan (a mix of green and blue).
- **Qt::magenta**: Represents the color magenta (a mix of red and blue).
- **Qt::yellow**: Represents the color yellow.
- **Qt::darkRed**: Represents a dark shade of red.
- **Qt::darkGreen**: Represents a dark shade of green.
- **Qt::darkBlue**: Represents a dark shade of blue.
- **Qt::darkCyan**: Represents a dark shade of cyan.
- **Qt::darkMagenta**: Represents a dark shade of magenta.
- **Qt::darkYellow**: Represents a dark shade of yellow.
- **Qt::gray**: Represents the color gray.
- **Qt::darkGray**: Represents a dark shade of gray.
- **Qt::lightGray**: Represents a light shade of gray.
- **Qt::transparent**: Represents a transparent color.

QPainter Methods

- **begin(QPaintDevice *device)** $\mapsto$ `bool`: Initializes the painter for the given paint device.
- **end()** $\mapsto$ `bool`: Ends the painting process and releases any resources used for painting.
- **pen() const** $\mapsto$ `QPen`: Returns the currently used pen.
- **brush() const** $\mapsto$ `QBrush`: Returns the currently used brush.
- **setPen(const QPen &pen)**: Sets the pen to be used for drawing lines and outlines.
- **setBrush(const QBrush &brush)**: Sets the brush to be used for filling shapes.
- **setFont(const QFont &font)**: Sets the font to be used for drawing text.
- **font() const** $\mapsto$ `QFont`: Returns the font currently set for the QPainter.
- **save()**: Saves the current state of the painter.
- **restore()**: Restores the painter to the state saved by the most recent call to save().
- **setTransform(const QTransform &transform, bool combine = false)**: Sets the transformation matrix for the painter.
- **transform() const** $\mapsto$ `QTransform`: Returns the current transformation matrix.
- **setRenderHint(QPainter::RenderHint hint, bool on = true)**: Sets a render hint to improve drawing quality.
- **drawLine(const QPoint &p1, const QPoint &p2)**: Draws a line between the points p1 and p2.
- **drawRect(const QRect &rect)**: Draws a rectangle with the top-left corner and size specified by rect.
- **drawEllipse(const QRect &rect)**: Draws an ellipse inside the specified rectangular area.
- **drawText(const QPoint &point, const QString &text)**: Draws the given text at the specified point.
- **drawPixmap(const QPoint &point, const QPixmap &pixmap)**: Draws a pixmap at the given point.
- **drawImage(const QPoint &point, const QImage &image)**: Draws an image at the specified point.
- **fillRect(const QRect &rect, const QBrush &brush)**: Fills the given rectangle with the specified brush.
- **translate(const QPointF &offset)**: Translates the coordinate system by the given offset.
- **rotate(qreal angle)**: Rotates the coordinate system by the specified angle.
- **scale(qreal sx, qreal sy)**: Scales the coordinate system by the factors sx and sy.

Shape Objects <QtCore>

- **QPoint**: Represents a point in 2D space with integer precision.
- **QPointF**: Represents a point in 2D space with floating-point precision.
- **QSize**: Defines the size of a 2D object using integer precision.
- **QSizeF**: Defines the size of a 2D object using floating-point precision.
- **QRect**: Represents a rectangle in 2D space with integer precision.
- **QRectF**: Represents a rectangle in 2D space with floating-point precision.
- **QLine**: Represents a line in 2D space with integer precision.
- **QLineF**: Represents a line in 2D space with floating-point precision.
- **QPolygon**: Represents a polygon defined by a vector of points with integer precision.
- **QPolygonF**: Represents a polygon defined by a vector of points with floating-point precision.
- **QRegion**: Represents a region in a plane, which can be non-contiguous.
- **QPath**: Represents a path, potentially containing lines, curves, and subpaths.
- **QPainterPath**: Represents a path that can be drawn with QPainter, including moveto, lineto, and curveto operations.

Layout Methods

17.1 QHBoxLayout Methods

- **addStretch(int stretch = 0)**: Adds a stretchable space to the layout.
- **addWidget(QWidget *widget, int stretch = 0, Qt::Alignment alignment = 0)**: Adds a widget to the layout.
- **insertStretch(int index, int stretch = 0)**: Inserts a stretchable space at the specified index in the layout.
- **insertWidget(int index, QWidget *widget, int stretch = 0, Qt::Alignment alignment = 0)**: Inserts a widget at the specified index in the layout.

18 QVBoxLayout Methods

- **addStretch(int stretch = 0)**: Adds a stretchable space to the layout.
- **addWidget(QWidget *widget, int stretch = 0, Qt::Alignment alignment = 0)**: Adds a widget to the layout.
- **insertStretch(int index, int stretch = 0)**: Inserts a stretchable space at the specified index in the layout.
- **insertWidget(int index, QWidget *widget, int stretch = 0, Qt::Alignment alignment = 0)**: Inserts a widget at the specified index in the layout.

18.1 QGridLayout Methods

- **addWidget(QWidget *widget, int row, int column, Qt::Alignment alignment = 0)**: Adds a widget to the layout at the specified row and column.
- **addWidget(QWidget *widget, int row, int column, int rowSpan, int columnSpan, Qt::Alignment alignment = 0)**: Adds a widget to the layout, spanning multiple rows and/or columns.
- **setRowStretch(int row, int stretch)**: Sets the stretch factor for the specified row.
- **setColumnStretch(int column, int stretch)**: Sets the stretch factor for the specified column.
- **setRowMinimumHeight(int row, int minSize)**: Sets the minimum height for the specified row.
- **setColumnMinimumWidth(int column, int minSize)**: Sets the minimum width for the specified column.

Other QObjects derived objects

19.1 Widget Classes

- **QWidget**: Base class for all UI objects.
- **QMainWindow**: Main application window class.
- **QDialog**: Base class for dialog windows.
- **QFrame**: Frame widget class.
- **QLineEdit**: Single-line text input widget.
- **QTextEdit**: Rich text editing widget.
- **QListView**, **QTableView**, **QTreeView**: For displaying data in list, table, and tree formats.
- **QComboBox**: Combines a button with a dropdown list.

19.2 Layout Classes

- **QLayout**: Base class for layouts.
- **QHBoxLayout**, **QVBoxLayout**: Horizontal and vertical box layouts.
- **QGridLayout**: Grid layout class.
- **QStackedLayout**: Stacking widgets layout.

19.3 Event and IO Classes

- **QTimer**: Timer class.
- **QEventLoop**: Event loop manager.
- **QIODevice**: Base class for IO devices.
- **QFile**, **QDataStream**, **QTextStream**: File and data stream classes.

19.4 Networking Classes

- **QNetworkAccessManager**: Network operations class.
- **QTcpSocket**, **QUdpSocket**: TCP and UDP socket classes.
- **QNetworkRequest**, **QNetworkReply**: Network request and response classes.

19.5 Model/View Classes

- **QAbstractItemModel**, **QStandardItemModel**: Base classes for item models.
- **QAbstractListModel**, **QAbstractTableModel**: List and table model classes.

19.6 Graphics View Framework

- **QGraphicsItem**, **QGraphicsScene**, **QGraphicsView**: Classes for 2D graphics.

19.7 Multimedia Classes

- **QMediaPlayer**, **QAudioOutput**: Multimedia handling classes.

19.8 Utility Classes

- **QProcess**: External program execution class.
- **QThread**: Threading support class.

19.9 Other Core Classes

- **QApplication**: Manages application-wide resources.
- **QObject**: Base class for many Qt classes.

Other QWidget derived objects

- **QMainWindow**: Main window class, providing a framework for building an application's main user interface.
- **QDialog**: Base class for dialog windows, used for creating modal or modeless dialogs.
- **QFrame**: Frame class, used to provide a frame and a background for other widgets.
- **QLabel**: Label widget, used for displaying text or images.
- **QPushButton**: Button widget, commonly used for receiving user inputs like clicks.
- **QLineEdit**: Single-line text editing widget, allowing user input of text strings.
- **QTextEdit**: Rich text editing widget, capable of displaying and editing formatted text.
- **QComboBox**: Combines a line edit for editing and a dropdown list for selecting text items.
- **QCheckBox**: Checkbox widget, providing an option that can be checked or unchecked.
- **QRadioButton**: Radio button widget, used for selecting one of a set of options.
- **QSlider**: Slider widget, used for selecting a value from a range.
- **QSpinBox**: Spin box widget, allowing selection of a value from a range of values.
- **QTabWidget**: Tab widget, used for stacking multiple widgets and allowing navigation between them via tabs.
- **QListWidget**, **QTreeWidget**, **QTableWidget**: High-level item-based widgets for displaying lists, trees, and tables of data.
- **QGroupBox**: Group box widget, used to group collections of widgets.
- **QToolBar**: Toolbar widget, providing a set of tool buttons.
- **QStatusBar**: Status bar widget, used for displaying status information.
- **QProgressBar**: Progress bar widget, for giving feedback about the progress of a task.
- **QGraphicsView**: Widget for displaying contents of a QGraphicsScene.