Contents

Qt Style Sheets Reference

Qt Style Sheets support various properties, pseudo-states, and subcontrols that make it possible to customize the look of widgets.

List of Stylable Widgets

The following table lists the Qt widgets that can be customized using style sheets:

Widget	How to Style
QAbstractScrollArea	Supports the box model. All derivatives of QAbstractScrollArea, including QTextEdit, and QAbstractItemView (all item view classes), support scrollable backgrounds using background-attachment. Setting the background-attachment to fixed provides a background-image that does not scroll with the viewport. Setting the background-attachment to scroll, scrolls the background-image when the scroll bars move. See Customizing QAbstractScrollArea for an example.
QCheckBox	Supports the box model. The check indicator can be styled using the ::indicator subcontrol. By default, the indicator is placed in the Top Left corner of the Contents rectangle of the widget. The spacing property specifies the spacing between the check indicator and the text. See Customizing QCheckBox for an example.
QColumnView	The grip can be styled be using the image property. The arrow indicators can by styled using the ::left-arrow subcontrol and the ::right-arrow subcontrol.
QComboBox	The frame around the combobox can be styled using the box model. The drop-down button can be styled using the ::drop-down subcontrol. By default, the drop-down button is placed in the top right corner of the padding rectangle of the widget. The arrow mark inside the drop-down button can be styled using the ::down-arrow subcontrol. By default, the arrow is placed in the center of the contents rectangle of the drop-down subcontrol. See Customizing QComboBox for an example.
QDateEdit	See QSpinBox.
QDateTimeEdit	See QSpinBox.
QDialog	Supports only the background, background-clip and background-origin properties. **Warning:** Make sure you define the Q_OBJECT macro for your custom widget.
QDialogButtonBox	The layout of buttons can be altered using the button-layout property.
QDockWidget	Supports styling of the title bar and the title bar buttons when docked. The dock widget border can be styled using the border property. The ::title subcontrol can be used to customize the title bar. The close and float buttons are positioned with respect to the ::title subcontrol using the ::close-button and ::float-button respectively. When the title bar is vertical, the :vertical pseudo class is set. In addition, depending on QDockWidget::DockWidgetFeature, the :closable, :floatable and :movable pseudo states are set. **Note:** Use QMainWindow::separator to style the resize handle. **Warning:** The style sheet has no effect when the QDockWidget is undocked as Qt uses native top level windows when undocked. See Customizing QDockWidget for an example.
QDoubleSpinBox	See QSpinBox.
QFrame	Supports the box model. Since 4.3, setting a stylesheet on a QLabel automatically sets the QFrame::frameStyle property to QFrame::StyledPanel. See Customizing QFrame for an example.
QGroupBox	Supports the box model. The title can be styled using the ::title subcontrol. By default, the title is placed depending on QGroupBox::textAlignment.

	In the case of a checkable QGroupBox, the title includes the check indicator. The indicator is styled using the the ::indicator subcontrol. The spacing property can be used to control the spacing between the text and indicator. See Customizing QGroupBox for an example.
QHeaderView	Supports the box model. The sections of the header view are styled using the ::section sub control. The section Sub-control supports the :middle, :first, :last, :only-one, :next-selected, :previous-selected, :selected, and :checked pseudo states. Sort indicator in can be styled using the ::up-arrow and the ::down-arrow Sub-control. See Customizing QHeaderView for an example.
QLabel	Supports the box model. Does not support the :hover pseudo-state. Since 4.3, setting a stylesheet on a QLabel automatically sets the QFrame::frameStyle property to QFrame::StyledPanel. See Customizing QFrame for an example (a QLabel derives from QFrame).
QLineEdit	Support the box model. The color and background of the selected item is styled using selection-color and selection-background-color respectively. The password character can be styled using the lineedit-password-character property. See Customizing QLineEdit for an example.
QListView	Supports the box model. When alternating row colors is enabled, the alternating colors can be styled using the alternate-background-color property. The color and background of the selected item is styled using selection-color and selection-background-color respectively. The selection behavior is controlled by the show-decoration-selected property. Use the ::item subcontrol for more fine grained control over the items in the QListView. See QAbsractScrollArea to style scrollable backgrounds. See Customzing QListView for an example.
QListWidget	See QListView.
QMainWindow	Supports styling of the separator The separator in a QMainWindow when using QDockWidget is styled using the ::separator subcontrol. See Customizing QMainWindow for an example.
QMenu	Supports the box model. Individual items are styled using the ::item subcontrol. In addition to the usually supported pseudo states, item subcontrol supports the :selected, :default, :exclusive and the non-exclusive pseudo states. The indicator of checkable menu items is styled using the ::indicator subcontrol. The separator is styled using the ::separator subcontrol. For items with a sub menu, the arrow marks are styled using the right-arrow and left-arrow. The scroller is styled using the ::scroller. The tear-off is styled using the ::tearoff. See Customizing QMenu for an example.
QMenuBar	Supports the box model. The spacing property specifies the spacing between menu items. Individual items are styled using the ::item subcontrol. **Warning:** When running on Qt/Mac, the menu bar is usually embedded into the system-wide menu bar. In this case, the style sheet will have no effect. See Customizing QMenuBar for an example.
QMessageBox	The messagebox-text-interaction-flags property can be used to alter the interaction with text in the message box.
QProgressBar	Supports the box model. The chunks of the progress bar can be styled using the ::chunk subcontrol. The chunk is displayed on the Contents rectangle of the widget. If the progress bar displays text, use the text-align property to position the text. Indeterminate progress bars have the :indeterminate pseudo state set. See Customizing QProgressBar for an example.
QPushButton	Supports the box model. Supports the :default, :flat, :checked pseudo states. For QPushButton with a menu, the menu indicator is styled using the ::menu-indicator subcontrol. Appearance of checkable push buttons can be customized using the :open and :closed pseudo-states. **Warning:** If you only set a background-color on a QPushButton, the background may not appear unless you set the border property to some value. This is because, by default, the QPushButton draws a native border which completely overlaps the background-color. For example, `QPushButton { background-color: red; border: none; }` See Customizing QPushButton for an example.
QRadioButton	Supports the box model. The check indicator can be styled using the ::indicator subcontrol. By default, the indicator is placed in the Top Left corner of the Contents rectangle of the widget. The spacing property specifies the spacing between the check indicator and the text. See Customizing QRadioButton for an example.
QScrollBar	Supports the box model. The Contents rectangle of the widget is considered to be the groove over which the slider moves. The extent of the QScrollBar (i.e the width or the height depending on the orientation) is set using the width or height property respectively. To determine the orientation, use the :horizontal and the :vertical pseudo states. The slider can be styled using the ::handle subcontrol. Setting the min-width or

	min-height provides size contraints for the slider depending on the orientation. The ::add-line subcontrol can be used to style the button to add a line. By default, the add-line subcontrol is placed in top right corner of the Border rectangle of the widget. Depending on the orientation the ::right-arrow or ::down-arrow. By default, the arrows are placed in the center of the Contents rectangle of the add-line subcontrol. The ::sub-line subcontrol can be used to style the button to subtract a line. By default, the sub-line subcontrol is placed in bottom right corner of the Border rectangle of the widget. Depending on the orientation the ::left-arrow or ::up-arrow. By default, the arrows are placed in the center of the Contents rectangle of the sub-line subcontrol. The ::sub-page subcontrol can be used to style the region of the slider that subtracts a page. The ::add-page subcontrol can be used to style the region of the slider that adds a page. See Customizing QScrollBar for an example.
QSizeGrip	Supports the width, height, and image properties. See Customizing QSizeGrip for an example.
QSlider	Supports the box model. For horizontal slides, the min-width and height properties must be provided. For vertical sliders, the min-height and width properties must be provided. The groove of the slider is styled using the ::groove. The groove is positioned by default in the Contents rectangle of the widget. The thumb of the slider is styled using ::handle subcontrol. The subcontrol moves in the Contents rectangle of the groove subcontrol. See Customizing QSlider for an example.
QSpinBox	The frame of the spin box can be styled using the box model. The up button and arrow can be styled using the ::up-button and ::up-arrow subcontrols. By default, the up-button is placed in the top right corner in the Padding rectangle of the widget. Without an explicit size, it occupies half the height of its reference rectangle. The up-arrow is placed in the center of the Contents rectangle of the up-button. The down button and arrow can be styled using the ::down-button and ::down-arrow subcontrols. By default, the down-button is placed in the bottom right corner in the Padding rectangle of the widget. Without an explicit size, it occupies half the height of its reference rectangle. The bottom-arrow is placed in the center of the Contents rectangle of the bottom-button. See Customizing QSpinBox for an example.
QSplitter	Supports the box model. The handle of the splitter is styled using the ::handle subcontrol. See Customizing QSplitter for an example.
QStatusBar	Supports only the background property. The frame for individual items can be style using the ::item subcontrol. See Customizing QStatusBar for an example.
QTabBar	Individual tabs may be styled using the ::tab subcontrol. Close buttons using the ::close-button The tabs support the :only-one, :first, :last, :middle, :previous--selected, :next-selected, :selected pseudo states. The :top, :left, :right, :bottom pseudo states depending on the orientation of the tabs. Overlapping tabs for the selected state are created by using negative margins or using the absolute position scheme. The tear indicator of the QTabBar is styled using the ::tear subcontrol. QTabBar used two QToolButtons for its scrollers that can be styled using the QTabBar QToolButton selector. To specify the width of the scroll button use the ::scroller subcontrol. The alignment of the tabs within the QTabBar is styled using the alignment property. **Warning:** To change the position of the QTabBar within a QTabWidget, use the tab-bar subcontrol (and set subcontrol-position). See Customizing QTabBar for an example.
QTabWidget	The frame of the tab widget is styled using the ::pane subcontrol. The left and right corners are styled using the ::left-corner and ::right-corner respectively. The position of the tab bar is controlled using the ::tab-bar subcontrol. By default, the subcontrols have positions of a QTabWidget in the QWindowsStyle. To place the QTabBar in the center, set the subcontrol-position of the tab-bar subcontrol. The :top, :left, :right, :bottom pseudo states depending on the orientation of the tabs. See Customizing QTabWidget for an example.
QTableView	Supports the box model. When alternating row colors is enabled, the alternating colors can be styled using the alternate-background-color property. The color and background of the selected item is styled using selection-color and selection-background-color respectively. The corner widget in a QTableView is implemented as a QAbstractButton and can be styled using the "QTableView QTableCornerButton::section" selector. **Warning:** If you only set a background-color on a QTableCornerButton, the background may not appear unless you set the border property to some value. This is because, by default, the QTableCornerButton draws a native border which completely overlaps the background-color. The color of the grid can be specified using the gridline-color property. See QAbsractScrollArea to style scrollable backgrounds. See Customzing QTableView for an example.
QTableWidget	See QTableView.
QTextEdit	Supports the box model. The color and background of selected text is styled using selection-color and selection-background-color respectively.

	See QAbsractScrollArea to style scrollable backgrounds.
QTimeEdit	See QSpinBox.
QToolBar	Supports the box model. The :top, :left, :right, :bottom pseudo states depending on the area in which the tool bar is grouped. The :first, :last, :middle, :only-one pseudo states indicator the position of the tool bar within a line group (See QStyleOptionToolBar::positionWithinLine). The separator of a QToolBar is styled using the ::separator subcontrol. The handle (to move the toolbar) is styled using the ::handle subcontrol. See Customizing QToolBar for an example.
QToolButton	Supports the box model. If the QToolButton has a menu, is ::menu-indicator subcontrol can be used to style the indicator. By default, the menu-indicator is positioned at the bottom right of the Padding rectangle of the widget. If the QToolButton is in QToolButton::MenuButtonPopup mode, the ::menu-button subcontrol is used to draw the menu button. ::menu-arrow subcontrol is used to draw the menu arrow inside the menu-button. By default, it is positioned in the center of the Contents rectangle of the menu-button subcontrol. When the QToolButton displays arrows, the ::up-arrow, ::down-arrow, ::left-arrow and ::right-arrow subcontrols are used. **Warning:** If you only set a background-color on a QToolButton, the background will not appear unless you set the border property to some value. This is because, by default, the QToolButton draws a native border which completely overlaps the background-color. For example, `QToolButton { background-color: red; border: none; }` See Customizing QToolButton for an example.
QToolBox	Supports the box model. The individual tabs can by styled using the ::tab subcontrol. The tabs support the :only-one, :first, :last, :middle, :previous-selected, :next-selected, :selected pseudo states.
QToolTip	Supports the box model. The opacity property controls the opacity of the tooltip. See Customizing QFrame for an example (a QToolTip is a QFrame).
QTreeView	Supports the box model. When alternating row colors is enabled, the alternating colors can be styled using the alternate-background-color property. The color and background of the selected item is styled using selection-color and selection-background-color respectively. The selection behavior is controlled by the show-decoration-selected property. The branches of the tree view can be styled using the ::branch subcontrol. The ::branch Sub-control supports the :open, :closed, :has-sibling and :has-children pseudo states. Use the ::item subcontrol for more fine grained control over the items in the QTreeView. See QAbsractScrollArea to style scrollable backgrounds. See Customizing QTreeView for an example to style the branches.
QTreeWidget	See QTreeView.

QWidget	Supports only the background, background-clip and background-origin properties.
	If you subclass from QWidget, you need to provide a paintEvent for your custom QWidget as below:
	`void CustomWidget::paintEvent(QPaintEvent *)`
	`{`
	`  QStyleOption opt;`
	`  opt.init(this);`
	`  QPainter p(this);`
	`  style()->drawPrimitive(QStyle::PE_Widget, &opt, &p, this);`
	`}`
	The above code is a no-operation if there is no stylesheet set.
	Warning: Make sure you define the Q_OBJECT macro for your custom widget.

List of Properties

The table below lists all the properties supported by Qt Style Sheets. Which values can be given to an property depend on the property's type. Unless otherwise specified, properties below apply to all widgets. Properties marked with an asterisk * are specific to Qt and have no equivalent in CSS2 or CSS3.

Property	Type	Description

alternate-background-color	Brush	The alternate background color used in QAbstractItemView subclasses.
		If this property is not set, the default value is whatever is set for the palette's AlternateBase role.
		Example:
		`QTreeView {`
		`  alternate-background-color: blue;`
		`  background: yellow;`
		`}`
		See also background and selection-background-color.

background	Background	Shorthand notation for setting the background. Equivalent to specifying background-color, background-image, background-repeat, and/or background-position. This property is supported by QAbstractItemView subclasses, QAbstractSpinBox subclasses, QCheckBox, QComboBox, QDialog, QFrame,

		QGroupBox, QLabel, QLineEdit, QMenu, QMenuBar, QPushButton, QRadioButton, QSplitter, QTextEdit, QToolTip, and plain QWidgets.
		Example:
		`QTextEdit { background: yellow }`
		Often, it is required to set a fill pattern similar to the styles in Qt::BrushStyle. You can use the background-color property for Qt::SolidPattern, Qt::RadialGradientPattern, Qt::LinearGradientPattern and Qt::ConicalGradientPattern. The other patterns are easily achieved by creating a background image that contains the pattern.
		Example:
		`QLabel { background-image: url(dense6pattern.png); background-repeat: repeat-xy; }`
		See also background-origin, selection-background-color, background-clip, background-attachment and alternate-background-color.

background-color	Brush	The background color used for the widget.
		Examples:
		`QLabel { background-color: yellow }`
		`QLineEdit { background-color: rgb(255, 0, 0) }`

background-image	Url	The background image used for the widget. Semi-transparent parts of the image let the background-color shine through. Example: `QFrame { background-image: url(:/images/hydro.png) }`
background-repeat	Repeat	Whether and how the background image is repeated to fill the background-origin rectangle. If this property is not specified, the background image is repeated in both directions (repeat). Example: `QFrame { background: white url(:/images/ring.png); background-repeat: repeat-y; background-position: left; }`
background-position	Alignment	The alignment of the background image within the background-origin rectangle. If this property is not specified, the alignment is top left. Example: `QFrame { background: url(:/images/footer.png); background-position: bottom left; }`
background-attachment	Attachment	Determines whether the background-image in a QAbstractScrollArea is scrolled or fixed with respect to the viewport. By default, the background-image scrolls with the viewport. Example: `QTextEdit { background-image: url("leaves.png"); background-attachment: fixed; }` See also background
background-clip	Origin	The widget's rectangle, in which the background is drawn. This property specifies the rectangle to which the background-color and background-image are clipped. This property is supported by QAbstractItemView subclasses, QAbstractSpinBox subclasses, QCheckBox, QComboBox, QDialog, QFrame, QGroupBox, QLabel, QPushButton, QRadioButton, QSplitter, QTextEdit, QToolTip, and plain QWidgets. If this property is not specified, the default is border. Example: `QFrame { background-image: url(:/images/header.png); background-position: top left; background-origin: content; background-clip: padding; }` See also background, background-origin and The Box Model.
background-origin	Origin	The widget's background rectangle, to use in conjunction with background-position and background-image. This property is supported by QAbstractItemView subclasses, QAbstractSpinBox subclasses, QCheckBox, QComboBox, QDialog, QFrame, QGroupBox, QLabel, QPushButton, QRadioButton, QSplitter, QTextEdit, QToolTip, and plain QWidgets. If this property is not specified, the default is padding.

		Example: `QFrame { background-image: url(:/images/header.png); background-position: top left; background-origin: content; }` See also background and The Box Model.
border	Border	Shorthand notation for setting the widget's border. Equivalent to specifying border-color, border-style, and/or border-width. This property is supported by QAbstractItemView subclasses, QAbstractSpinBox subclasses, QCheckBox, QComboBox, QFrame, QGroupBox, QLabel, QLineEdit, QMenu, QMenuBar, QPushButton, QRadioButton, QSplitter, QTextEdit, QToolTip, and plain QWidgets. Example: `QLineEdit { border: 1px solid white }`
border-top	Border	Shorthand notation for setting the widget's top border. Equivalent to specifying border-top-color, border-top-style, and/or border-top-width.
border-right	Border	Shorthand notation for setting the widget's right border. Equivalent to specifying border-right-color, border-right-style, and/or border-right-width.
border-bottom	Border	Shorthand notation for setting the widget's bottom border. Equivalent to specifying border-bottom-color, border-bottom-style, and/or border-bottom-width.
border-left	Border	Shorthand notation for setting the widget's left border. Equivalent to specifying border-left-color, border-left-style, and/or border-left-width.
border-color	Box Colors	The color of all the border's edges. Equivalent to specifying border-top-color, border-right-color, border-bottom-color, and border-left-color. This property is supported by QAbstractItemView subclasses, QAbstractSpinBox subclasses, QCheckBox, QComboBox, QFrame, QGroupBox, QLabel, QLineEdit, QMenu, QMenuBar, QPushButton, QRadioButton, QSplitter, QTextEdit, QToolTip, and plain QWidgets. If this property is not specified, it defaults to color (i.e., the widget's foreground color). Example: `QLineEdit { border-width: 1px; border-style: solid; border-color: white; }` See also border-style, border-width, border-image, and The Box Model.
border-top-color	Brush	The color of the border's top edge.
border-right-color	Brush	The color of the border's right edge.
border-bottom-color	Brush	The color of the border's bottom edge.
border-left-color	Brush	The color of the border's left edge.
border-image	Border Image	The image used to fill the border. The image is cut into nine parts and stretched appropriately if necessary. See Border Image for details. This property is supported by QAbstractItemView subclasses, QAbstractSpinBox subclasses, QCheckBox, QComboBox, QFrame, QGroupBox, QLabel, QLineEdit, QMenu, QMenuBar, QPushButton, QRadioButton, QSplitter, QTextEdit and QToolTip. See also border-color, border-style, border-width, and The Box Model.
border-radius	Radius	The radius of the border's corners. Equivalent to specifying border-top-left-radius, border-top-right-radius, border-bottom-right-radius, and border-bottom-left-radius. The border-radius clips the element's background. This property is supported by QAbstractItemView subclasses, QAbstractSpinBox subclasses, QCheckBox, QComboBox, QFrame, QGroupBox, QLabel, QLineEdit, QMenu, QMenuBar, QPushButton, QRadioButton, QSplitter, QTextEdit, and QToolTip. If this property is not specified, it defaults to 0. Example: `QLineEdit { border-width: 1px; border-style: solid; border-radius: 4px; }` See also border-width and The Box Model.
border-top-left-radius	Radius	The radius of the border's top-left corner.
border-top-right-radius	Radius	The radius of the border's top-right corner.
border-bottom-right-radius	Radius	The radius of the border's bottom-right corner. Setting this property to a positive value results in a rounded corner.

border-bottom-left-radius	Radius	The radius of the border's bottom-left corner. Setting this property to a positive value results in a rounded corner.
border-style	Border Style	The style of all the border's edges. This property is supported by QAbstractItemView subclasses, QAbstractSpinBox subclasses, QCheckBox, QComboBox, QFrame, QGroupBox, QLabel, QLineEdit, QMenu, QMenuBar, QPushButton, QRadioButton, QSplitter, QTextEdit, and QToolTip. If this property is not specified, it defaults to none. Example: `QLineEdit { border-width: 1px; border-style: solid; border-color: blue; }` See also border-color, border-style, border-image, and The Box Model.
border-top-style	Border Style	The style of the border's top edge.
border-right-style	Border Style	The style of the border's right edge/
border-bottom-style	Border Style	The style of the border's bottom edge.
border-left-style	Border Style	The style of the border's left edge.
border-width	Box Lengths	The width of the border. Equivalent to setting border-top-width, border-right-width, border-bottom-width, and border-left-width. This property is supported by QAbstractItemView subclasses, QAbstractSpinBox subclasses, QCheckBox, QComboBox, QFrame, QGroupBox, QLabel, QLineEdit, QMenu, QMenuBar, QPushButton, QRadioButton, QSplitter, QTextEdit, and QToolTip. Example: `QLineEdit { border-width: 2px; border-style: solid; border-color: darkblue; }` See also border-color, border-radius, border-style, border-image, and The Box Model.
border-top-width	Length	The width of the border's top edge.
border-right-width	Length	The width of the border's right edge.
border-bottom-width	Length	The width of the border's bottom edge.
border-left-width	Length	The width of the border's left edge.
bottom	Length	If position is relative (the default), moves a subcontrol by a certain offset up; specifying bottom: *y* is then equivalent to specifying top: -*y*. If position is absolute, the bottom property specifies the subcontrol's bottom edge in relation to the parent's bottom edge (see also subcontrol-origin). Example: `QSpinBox::down-button { bottom: 2px }` See also left, right, and top.
button-layout	Number	The layout of buttons in a QDialogButtonBox or a QMessageBox. The possible values are 0 (WinLayout), 1 (MacLayout), 2 (KdeLayout), and 3 (GnomeLayout). If this property is not specified, it defaults to the value specified by the current style for the SH_DialogButtonLayout style hint. Example: `* { button-layout: 2 }`
color	Brush	The color used to render text. This property is supported by all widgets that respect the QWidget::palette. If this property is not set, the default is whatever is set for in the widget's palette for the QWidget::foregroundRole (typically black). Example: `QPushButton { color: red }` See also background and selection-color.
dialogbuttonbox-buttons-have-icons	Boolean	Whether the buttons in a QDialogButtonBox show icons If this property is set to 1, the buttons of a QDialogButtonBox show icons; if it is set to 0, the icons are not shown. See the List of Icons section for information on how to set icons. `QDialogButtonBox { dialogbuttonbox-buttons-have-icons: 1; }` **Note:** Styles defining this property must be applied before the QDialogButtonBox is created; this means that you must apply the style to

		the parent widget or to the application itself.
font	Font	Shorthand notation for setting the text's font. Equivalent to specifying font-family, font-size, font-style, and/or font-weight. This property is supported by all widgets that respect the QWidget::font. If this property is not set, the default is the QWidget::font. Example: `QCheckBox { font: bold italic large "Times New Roman" }`
font-family	String	The font family. Example: `QCheckBox { font-family: "New Century Schoolbook" }`
font-size	Font Size	The font size. In this version of Qt, only pt and px metrics are supported. Example: `QTextEdit { font-size: 12px }`
font-style	Font Style	The font style. Example: `QTextEdit { font-style: italic }`
font-weight	Font Weight	The weight of the font.
gridline-color*	Color	The color of the grid line in a QTableView. If this property is not specified, it defaults to the value specified by the current style for the SH_Table_GridLineColor style hint. Example: `* { gridline-color: gray }`
height	Length	The height of a subcontrol (or in some case, a widget). If this property is not specified, it defaults to a value that depends on the subcontrol/widget and on the current style. **Warning:** Unless otherwise specified, this property has no effect when set on widgets. If you want a widget with a fixed height, set the min-height and max-height to the same value. Example: `QSpinBox::down-button { height: 10px }` See also width.
icon-size	Length	The width and height of the icon in a widget. The icon size of the following widgets can be set using this property. QCheckBox QListView QPushButton QRadioButton QTabBar QToolBar QToolBox QTreeView

image*	Url+	The image that is drawn in the contents rectangle of a subcontrol.
		The image property accepts a list of Urls or an svg. The actual image that is drawn is determined using the same algorithm as QIcon (i.e) the image is never scaled up but always scaled down if necessary. If a svg is specified, the image is scaled to the size of the contents rectangle. Setting the image property on sub controls implicitly sets the width and height of the sub-control (unless the image in a SVG).
		In Qt 4.3 and later, the alignment of the image within the rectangle can be specified using image-position.
		This property is for subcontrols only--we don't support it for other elements.
		Warning: The QIcon SVG plugin is needed to render SVG images.
		Example:
		`/* implicitly sets the size of down-button to the size of spindown.png */`
		`QSpinBox::down-button { image: url(:/images/spindown.png) }`

image-position	alignment	In Qt 4.3 and later, the alignment of the image image's position can be specified using relative or absolute position.
left	Length	If position is relative (the default), moves a subcontrol by a certain offset to the right. If position is absolute, the left property specifies the subcontrol's left edge in relation to the parent's left edge (see also subcontrol-origin). If this property is not specified, it defaults to 0. Example: `QSpinBox::down-button { left: 2px }`

		See also right, top, and bottom.
lineedit-password-character*	Number	The QLineEdit password character as a Unicode number. If this property is not specified, it defaults to the value specified by the current style for the SH_LineEdit_PasswordCharacter style hint. Example: `* { lineedit-password-character: 9679 }`
margin	Box Lengths	The widget's margins. Equivalent to specifying margin-top, margin-right, margin-bottom, and margin-left. This property is supported by QAbstractItemView subclasses, QAbstractSpinBox subclasses, QCheckBox, QComboBox, QFrame, QGroupBox, QLabel, QLineEdit, QMenu, QMenuBar, QPushButton, QRadioButton, QSplitter, QTextEdit, and QToolTip. If this property is not specified, it defaults to 0. Example: `QLineEdit { margin: 2px }` See also padding, spacing, and The Box Model.
margin-top	Length	The widget's top margin.
margin-right	Length	The widget's right margin.
margin-bottom	Length	The widget's bottom margin.
margin-left	Length	The widget's left margin.
max-height	Length	The widget's or a subcontrol's maximum height. This property is supported by QAbstractItemView subclasses, QAbstractSpinBox subclasses, QCheckBox, QComboBox, QFrame, QGroupBox, QLabel, QLineEdit, QMenu, QMenuBar, QPushButton, QRadioButton, QSizeGrip, QSpinBox, QSplitter, QStatusBar, QTextEdit, and QToolTip. The value is relative to the contents rect in the box model. Example: `QSpinBox { max-height: 24px }` See also max-width.
max-width	Length	The widget's or a subcontrol's maximum width. This property is supported by QAbstractItemView subclasses, QAbstractSpinBox subclasses, QCheckBox, QComboBox, QFrame, QGroupBox, QLabel, QLineEdit, QMenu, QMenuBar, QPushButton, QRadioButton, QSizeGrip, QSpinBox, QSplitter, QStatusBar, QTextEdit, and QToolTip. The value is relative to the contents rect in the box model. Example: `QComboBox { max-width: 72px }` See also max-height.
messagebox-text-interaction-flags*	Number	The interaction behavior for text in a message box. Possible values are based on Qt::TextInteractionFlags. If this property is not specified, it defaults to the value specified by the current style for the SH_MessageBox_TextInteractionFlags style hint. Example: `QMessageBox { messagebox-text-interaction-flags: 5 }`
min-height	Length	The widget's or a subcontrol's minimum height. This property is supported by QAbstractItemView subclasses, QAbstractSpinBox subclasses, QCheckBox, QComboBox, QFrame, QGroupBox, QLabel, QLineEdit, QMenu, QMenuBar, QPushButton, QRadioButton, QSizeGrip, QSpinBox, QSplitter, QStatusBar, QTextEdit, and QToolTip. If this property is not specified, the minimum height is derived based on the widget's contents and the style. The value is relative to the contents rect in the box model. Example: `QComboBox { min-height: 24px }` See also min-width.
min-width	Length	The widget's or a subcontrol's minimum width. This property is supported by QAbstractItemView subclasses, QAbstractSpinBox subclasses, QCheckBox, QComboBox, QFrame, QGroupBox, QLabel, QLineEdit, QMenu, QMenuBar, QPushButton, QRadioButton, QSizeGrip, QSpinBox, QSplitter, QStatusBar, QTextEdit, and QToolTip. If this property is not specified, the minimum width is derived based on the widget's contents and the style. The value is relative to the contents rect in the box model. Example: `QComboBox { min-width: 72px }` See also min-height.
opacity*	Number	The opacity for a widget. Possible values are from 0 (transparent) to 255 (opaque). For the moment, this is only supported for tooltips. If this property is not specified, it defaults to the value specified by the current style for the SH_ToolTipLabel_Opacity style hint. Example: `QToolTip { opacity: 223 }`
padding	Box	The widget's padding. Equivalent to specifying padding-top,

	Lengths	padding-right, padding-bottom, and padding-left. This property is supported by QAbstractItemView subclasses, QAbstractSpinBox subclasses, QCheckBox, QComboBox, QFrame, QGroupBox, QLabel, QLineEdit, QMenu, QMenuBar, QPushButton, QRadioButton, QSplitter, QTextEdit, and QToolTip. If this property is not specified, it defaults to 0. Example: `QLineEdit { padding: 3px }` See also margin, spacing, and The Box Model.
padding-top	Length	The widget's top padding.
padding-right	Length	The widget's right padding.
padding-bottom	Length	The widget's bottom padding.
padding-left	Length	The widget's left padding.
paint-alternating-row-colors-for-empty-area	bool	Whether the QTreeView paints alternating row colors for the empty area (i.e the area where there are no items)
position	relative | absolute	Whether offsets specified using left, right, top, and bottom are relative or absolute coordinates. If this property is not specified, it defaults to relative.
right	Length	If position is relative (the default), moves a subcontrol by a certain offset to the left; specifying right: *x* is then equivalent to specifying left: *-x*. If position is absolute, the right property specifies the subcontrol's right edge in relation to the parent's right edge (see also subcontrol-origin). Example: `QSpinBox::down-button { right: 2px }` See also left, top, and bottom.
selection-background-color*	Brush	The background of selected text or items. This property is supported by all widgets that respect the QWidget::palette and that show selection text. If this property is not set, the default value is whatever is set for the palette's Highlight role. Example: `QTextEdit { selection-background-color: darkblue }` See also selection-color and background.
selection-color*	Brush	The foreground of selected text or items. This property is supported by all widgets that respect the QWidget::palette and that show selection text. If this property is not set, the default value is whatever is set for the palette's HighlightedText role. Example: `QTextEdit { selection-color: white }` See also selection-background-color and color.
show-decoration-selected*	Boolean	Controls whether selections in a QListView cover the entire row or just the extent of the text. If this property is not specified, it defaults to the value specified by the current style for the SH_ItemView_ShowDecorationSelected style hint. Example: `* { show-decoration-selected: 1 }`
spacing*	Length	Internal spacing in the widget. This property is supported by QCheckBox, checkable QGroupBoxes, QMenuBar, and QRadioButton. If this property is not specified, the default value depends on the widget and on the current style. Example: `QMenuBar { spacing: 10 }` See also padding and margin.

subcontrol-origin*	Origin	The origin rectangle of the subcontrol within the parent element.
		If this property is not specified, the default is padding.
		Example:
		`QSpinBox::up-button {`
		`  image: url(:/images/spinup.png);`
		`  subcontrol-origin: content;`
		`  subcontrol-position: right top;`
		`}`
		See also subcontrol-position.

subcontrol-position*	Alignment	The alignment of the subcontrol within the origin rectangle specified by subcontrol-origin. If this property is not specified, it defaults to a value that depends on the subcontrol. Example:

```
QSpinBox::down-button {
    image: url(:/images/spindown.png);
    subcontrol-origin: padding;
    subcontrol-position: right bottom;
}
```

See also subcontrol-origin.

text-align	Alignment	The alignment of text and icon within the contents of the widget. If this value is not specified, it defaults to the value that depends on the native style. Example: `QPushButton { text-align: left; }` This property is currently supported only by QPushButton and QProgressBar.
text-decoration	none underline overline line-through	Additional text effects
top	Length	If position is relative (the default), moves a subcontrol by a certain offset down. If position is absolute, the top property specifies the subcontrol's top edge in relation to the parent's top edge (see also subcontrol-origin). If this property is not specified, it defaults to 0. Example: `QSpinBox::up-button { top: 2px }` See also left, right, and bottom.
width	Length	The width of a subcontrol (or a widget in some cases). If this property is not specified, it defaults to a value that depends on the subcontrol/widget and on the current style. **Warning:** Unless otherwise specified, this property has no effect when set on widgets. If you want a widget with a fixed width, set the min-width and max-width to the same value. Example: `QSpinBox::up-button { width: 12px }` See also height.

List of Icons

Icons used in Qt can be customized using the following properties. Each of the properties listed in this section have the type Icon.
Note that for icons to appear in buttons in a QDialogButtonBox, you need to set the dialogbuttonbox-buttons-have-icons property to true. Also, to customize the size of the icons, use the icon-size property.

Name	QStyle::StandardPixmap
backward-icon	QStyle::SP_ArrowBack
cd-icon	QStyle::SP_DriveCDIcon
computer-icon	QStyle::SP_ComputerIcon
desktop-icon	QStyle::SP_DesktopIcon
dialog-apply-icon	QStyle::SP_DialogApplyButton
dialog-cancel-icon	QStyle::SP_DialogCancelButton
dialog-close-icon	QStyle::SP_DialogCloseButton
dialog-discard-icon	QStyle::SP_DialogDiscardButton
dialog-help-icon	QStyle::SP_DialogHelpButton
dialog-no-icon	QStyle::SP_DialogNoButton
dialog-ok-icon	QStyle::SP_DialogOkButton
dialog-open-icon	QStyle::SP_DialogOpenButton
dialog-reset-icon	QStyle::SP_DialogResetButton
dialog-save-icon	QStyle::SP_DialogSaveButton
dialog-yes-icon	QStyle::SP_DialogYesButton
directory-closed-icon	QStyle::SP_DirClosedIcon
directory-icon	QStyle::SP_DirIcon
directory-link-icon	QStyle::SP_DirLinkIcon
directory-open-icon	QStyle::SP_DirOpenIcon
dockwidget-close-icon	QStyle::SP_DockWidgetCloseButton
downarrow-icon	QStyle::SP_ArrowDown
dvd-icon	QStyle::SP_DriveDVDIcon
file-icon	QStyle::SP_FileIcon
file-link-icon	QStyle::SP_FileLinkIcon
filedialog-contentsview-icon	QStyle::SP_FileDialogContentsView
filedialog-detailedview-icon	QStyle::SP_FileDialogDetailedView
filedialog-end-icon	QStyle::SP_FileDialogEnd
filedialog-infoview-icon	QStyle::SP_FileDialogInfoView
filedialog-listview-icon	QStyle::SP_FileDialogListView
filedialog-new-directory-icon	QStyle::SP_FileDialogNewFolder
filedialog-parent-directory-icon	QStyle::SP_FileDialogToParent

filedialog-start-icon	QStyle::SP_FileDialogStart
floppy-icon	QStyle::SP_DriveFDIcon
forward-icon	QStyle::SP_ArrowForward
harddisk-icon	QStyle::SP_DriveHDIcon
home-icon	QStyle::SP_DirHomeIcon
leftarrow-icon	QStyle::SP_ArrowLeft
messagebox-critical-icon	QStyle::SP_MessageBoxCritical
messagebox-information-icon	QStyle::SP_MessageBoxInformation
messagebox-question-icon	QStyle::SP_MessageBoxQuestion
messagebox-warning-icon	QStyle::SP_MessageBoxWarning
network-icon	QStyle::SP_DriveNetIcon
rightarrow-icon	QStyle::SP_ArrowRight
titlebar-contexthelp-icon	QStyle::SP_TitleBarContextHelpButton
titlebar-maximize-icon	QStyle::SP_TitleBarMaxButton
titlebar-menu-icon	QStyle::SP_TitleBarMenuButton
titlebar-minimize-icon	QStyle::SP_TitleBarMinButton
titlebar-normal-icon	QStyle::SP_TitleBarNormalButton
titlebar-shade-icon	QStyle::SP_TitleBarShadeButton
titlebar-unshade-icon	QStyle::SP_TitleBarUnshadeButton
trash-icon	QStyle::SP_TrashIcon
uparrow-icon	QStyle::SP_ArrowUp

List of Property Types

The following table summarizes the syntax and meaning of the different property types.

Type	Syntax	Description
Alignment	{ top | bottom | left | right | center }*	Horizontal and/or vertical alignment. Example: `QTextEdit { background-position: bottom center }`
Attachment	{ scroll | fixed }*	Scroll or fixed attachment.
Background	{ Brush | Url | Repeat | Alignment }*	A sequence of Brush, Url, Repeat, and Alignment.
Boolean	0 | 1	True (1) or false (0). Example: `QDialogButtonBox { dialogbuttonbox-buttons-have-icons: 1; }`
Border	{ Border Style | Length | Brush }*	Shorthand border property.
Border Image	none | Url Number{4} (stretch | repeat){0,2}	A border image is an image that is composed of nine parts (top left, top center, top right, center left, center, center right, bottom left, bottom center, and bottom right). When a border of a certain size is required, the corner parts are used as is, and the top, right, bottom, and left parts are stretched or repeated to produce a border with the desired size. See the CSS3 Draft Specification for details.
Border Style	dashed | dot-dash | dot-dot-dash | dotted | double | groove | inset | outset | ridge | solid | none	Specifies the pattern used to draw a border. See the CSS3 Draft Specification for details.

Box Colors	Brush{1,4}	One to four occurrences of Brush, specifying the top, right, bottom, and left edges of a box, respectively. If the left color is not specified, it is taken to be the same as the right color. If the bottom color is not specified, it is taken to be the same as the top color. If the right color is not specified, it is taken to be the same as the top color.
		Example:
		`QLabel { border-color: red }  /* red red red red */`
		`QLabel { border-color: red blue } /* red blue red blue */`
		`QLabel { border-color: red blue green } /* red blue green blue */`
		`QLabel { border-color: red blue green yellow }  /* red blue green yellow */`

Box Lengths	Length{1,4}	One to four occurrences of Length, specifying the top, right, bottom, and left edges of a box, respectively. If the left length is not specified, it is taken to be the same as the right length. If the bottom length is not specified, is it taken to be the same as the top length. If the right length is not specified, it is taken to be the same as the top length.
		Examples:
		`QLabel { border-width: 1px }  /* 1px 1px 1px 1px */`
		`QLabel { border-width: 1px 2px }  /* 1px 2px 1px 2px */`
		`QLabel { border-width: 1px 2px 3px }  /* 1px 2px 3px 2px */`
		`QLabel { border-width: 1px 2px 3px 4px }  /* 1px 2px 3px 4px */`

Brush	Color | Gradient | PaletteRole	Specifies a Color or a Gradient or an entry in the Palette.

Color	rgb(*r*, *g*, *b*)
	| rgba(*r*, *g*, *b*, *a*)
	| hsv(*h*, *s*, *v*)
	| hsva(*h*, *s*, *v*, *a*)
	| #*rrggbb*
	| Color Name
		Specifies a color as RGB (red, green, blue) or RGBA (red, green, blue, alpha) or HSV (hue, saturation, value) or HSVA (hue, saturation, value, alpha) or a named color. The rgb() or rgba() syntax can be used with integer values between 0 and 255, or with percentages. The value of s, v, and a in hsv() or hsva() must all be in the range 0-255; the value of h must be in the range 0-359.
		Examples:
		`QLabel { border-color: red }  /* opaque red */`
		`QLabel { border-color: #FF0000 }  /* opaque red */`
		`QLabel { border-color: rgba(255, 0, 0, 75%) }  /* 75% opaque red */`
		`QLabel { border-color: rgb(255, 0, 0) }  /* opaque red */`
		`QLabel { border-color: rgb(100%, 0%, 0%) }  /* opaque red */`
		`QLabel { border-color: hsv(60, 255, 255) }  /* opaque yellow */`
		`QLabel { border-color: hsva(240, 255, 255, 75%) }  /* 75% blue */`
		Note: The RGB colors allowed are the same as those allowed with CSS 2.1, as listed here.

Font	(Font Style | Font Weight) {0,2} Font Size String	Shorthand font property.
Font Size	Length	The size of a font.
Font Style	normal | italic | oblique	The style of a font.
Font Weight	normal | bold | 100 | 200 ... | 900	The weight of a font.

Gradient	qlineargradient
	|
	qradialgradient
	|
	qconicalgradient
		Specifies gradient fills. There are three types of gradient fills:
		Linear gradients interpolate colors between start and end points.
		Radial gradients interpolate colors between a focal point and end points on a circle surrounding it.
		Conical gradients interpolate colors around a center point.
		Gradients are specified in Object Bounding Mode. Imagine the box in which the gradient is rendered, to have its top left corner at (0, 0) and its bottom right corner at (1, 1). Gradient parameters are then specified as percentages from 0 to 1. These values are extrapolated to actual box coordinates at runtime. It is possible specify values that lie outside the bounding box (-0.6 or 1.8, for instance).
		Warning: The stops have to appear sorted in ascending order.
		Examples:
		`/* linear gradient from white to green */`
		`QTextEdit {`
		`  background: qlineargradient(x1:0, y1:0, x2:1, y2:1,`
		`  stop:0 white, stop: 0.4 gray, stop:1 green)`
		`}`
		`/* linear gradient from white to green */`
		`QTextEdit {`
		`  background: qlineargradient(x1:0, y1:0, x2:1, y2:1,`
		`  stop:0 white, stop: 0.4 rgba(10, 20, 30, 40), stop:1 rgb(0, 200, 230, 200))`
		`}`
		`/* conical gradient from white to green */`

```
QTextEdit {
    background: qconicalgradient(cx:0.5, cy:0.5, angle:30,
                stop:0 white, stop:1 #00FF00)
}

/* radial gradient from white to green */
QTextEdit {
    background: qradialgradient(cx:0, cy:0, radius: 1,
                fx:0.5, fy:0.5, stop:0 white, stop:1 green)
}
```

Icon	(Url (disabled | active | normal | selected)? (on | off)?)*	A list of url, QIcon::Mode and QIcon::State.
		Example:
		`* { file-icon: url(file.png), url(file_selected.png) selected; }`
		`QMessageBox { dialogbuttonbox-buttons-have-icons: true; dialog-ok-icon: url(ok.svg); dialog-cancel-icon: url(cancel.png), url(grayed_cancel.png) disabled; }`

Length	Number (px | pt | em | ex)?	A number followed by a measurement unit. The CSS standard recommends that user agents must ignore a declaration with an illegal value. In Qt, it is mandatory to specify measurement units. For compatibility with earlier versions of Qt, numbers without measurement units are treated as pixels in most contexts. The supported units are: px: pixels pt: the size of one point (i.e., 1/72 of an inch) em: the em width of the font (i.e., the width of 'M') ex: the ex width of the font (i.e., the height of 'x')
Number	A decimal integer or a real number	Examples: 0, 18, +127, -255, 12.34, -.5, 0009.
Origin	margin | border | padding | content	Indicates which of four rectangles to use. margin: The margin rectangle. The margin falls outside the border. border: The border rectangle. This is where any border is drawn. padding: The padding rectangle. Unlike the margins, padding is located inside the border. content: The content rectangle. This specifies where the actual contents go, excluding any padding, border, or margin. See also The Box Model.
PaletteRole	alternate-base | base | bright-text | button | button-text | dark | highlight | highlighted-text | light | link | link-visited | mid | midlight | shadow | text | window | window-text	These values correspond the Color roles in the widget's QPalette. For example, `QPushButton { color: palette(dark); }`
Radius	Length{1, 2}	One or two occurrences of Length. If only one length is specified, it is used as the radius of the quarter circle defining the corner. If two lengths are specified, the first length is the horizontal radius of a quarter ellipse, whereas the second length is the vertical radius.
Repeat	repeat-x | repeat-y | repeat | no-repeat	A value indicating the nature of repetition. repeat-x: Repeat horizontally. repeat-y: Repeat vertically. repeat: Repeat horizontally and vertically. no-repeat: Don't repeat.
Url	url(*filename*)	*filename* is the name of a file on the local disk or stored using the Qt Resource System. Setting an image implicitly sets the width and height of the element.

List of Pseudo-States

The following pseudo-states are supported:

Pseudo-State	Description
:active	This state is set when the widget resides in an active window.
:adjoins-item	This state is set when the ::branch of a QTreeView is adjacent to an item.
:alternate	This state is set for every alternate row whe painting the row of a QAbstractItemView when QAbstractItemView::alternatingRowColors() is set to true.
:bottom	The item is positioned at the bottom. For example, a QTabBar that has its tabs positioned at the bottom.
:checked	The item is checked. For example, the checked state of QAbstractButton.
:closable	The items can be closed. For example, the QDockWidget has the QDockWidget::DockWidgetClosable feature turned on.
:closed	The item is in the closed state. For example, an non-expanded item in a QTreeView
:default	The item is the default. For example, a default QPushButton or a default action in a QMenu.
:disabled	The item is disabled.
:editable	The QComboBox is editable.
:edit-focus	The item has edit focus (See QStyle::State_HasEditFocus). This state is available only for Qt Extended applications.
:enabled	The item is enabled.
:exclusive	The item is part of an exclusive item group. For example, a menu item in a exclusive QActionGroup.
:first	The item is the first (in a list). For example, the first tab in a QTabBar.
:flat	The item is flat. For example, a flat QPushButton.
:floatable	The items can be floated. For example, the QDockWidget has the QDockWidget::DockWidgetFloatable feature turned on.
:focus	The item has input focus.
:has-children	The item has children. For example, an item in a QTreeView that has child items.
:has-siblings	The item has siblings. For example, an item in a QTreeView that siblings.
:horizontal	The item has horizontal orientation
:hover	The mouse is hovering over the item.
:indeterminate	The item has indeterminate state. For example, a QCheckBox or QRadioButton is partially checked.
:last	The item is the last (in a list). For example, the last tab in a QTabBar.
:left	The item is positioned at the left. For example, a QTabBar that has its tabs positioned at the left.
:maximized	The item is maximized. For example, a maximized QMdiSubWindow.
:middle	The item is in the middle (in a list). For example, a tab that is not in the beginning or the end in a QTabBar.
:minimized	The item is minimized. For example, a minimized QMdiSubWindow.
:movable	The item can be moved around. For example, the QDockWidget has the QDockWidget::DockWidgetMovable feature turned on.
:no-frame	The item has no frame. For example, a frameless QSpinBox or QLineEdit.
:non-exclusive	The item is part of a non-exclusive item group. For example, a menu item in a non-exclusive QActionGroup.
:off	For items that can be toggled, this applies to items in the "off" state.
:on	For items that can be toggled, this applies to widgets in the "on" state.
:only-one	The item is the only one (in a list). For example, a lone tab in a QTabBar.
:open	The item is in the open state. For example, an expanded item in a QTreeView, or a QComboBox or QPushButton with an open menu.
:next-selected	The next item (in a list) is selected. For example, the selected tab of a QTabBar is next to this item.
:pressed	The item is being pressed using the mouse.
:previous-selected	The previous item (in a list) is selected. For example, a tab in a QTabBar that is next to the selected tab.
:read-only	The item is marked read only or non-editable. For example, a read only QLineEdit or a non-editable QComboBox.
:right	The item is positioned at the right. For example, a QTabBar that has its tabs positioned at the right.
:selected	The item is selected. For example, the selected tab in a QTabBar or the selected item in a QMenu.
:top	The item is positioned at the top. For example, a QTabBar that has its tabs positioned at the top.
:unchecked	The item is unchecked.
:vertical	The item has vertical orientation.
:window	The widget is a window (i.e top level widget)

List of Sub-Controls

The following subcontrols are available:

Sub-Control	Description

::add-line	The button to add a line of a QScrollBar.
::add-page	The region between the handle (slider) and the add-line of a QScrollBar.
::branch	The branch indicator of a QTreeView.
::chunk	The progress chunk of a QProgressBar.
::close-button	The close button of a QDockWidget or tabs of QTabBar
::corner	The corner between two scrollbars in a QAbstractScrollArea
::down-arrow	The down arrow of a QComboBox, QHeaderView (sort indicator), QScrollBar or QSpinBox.
::down-button	The down button of a QScrollBar or a QSpinBox.
::drop-down	The drop-down button of a QComboBox.
::float-button	The float button of a QDockWidget
::groove	The groove of a QSlider.
::indicator	The indicator of a QAbstractItemView, a QCheckBox, a QRadioButton, a checkable QMenu item or a checkable QGroupBox.
::handle	The handle (slider) of a QScrollBar, a QSplitter, or a QSlider.
::icon	The icon of a QAbstractItemView or a QMenu.
::item	An item of a QAbstractItemView, a QMenuBar, a QMenu, or a QStatusBar.
::left-arrow	The left arrow of a QScrollBar.
::left-corner	The left corner of a QTabWidget. For example, this control can be used to control position the left corner widget in a QTabWidget.
::menu-arrow	The arrow of a QToolButton with a menu.
::menu-button	The menu button of a QToolButton.
::menu-indicator	The menu indicator of a QPushButton.
::right-arrow	The right arrow of a QMenu or a QScrollBar.
::pane	The pane (frame) of a QTabWidget.
::right-corner	The right corner of a QTabWidget. For example, this control can be used to control the position the right corner widget in a QTabWidget.
::scroller	The scroller of a QMenu or QTabBar.
::section	The section of a QHeaderView.
::separator	The separator of a QMenu or in a QMainWindow.
::sub-line	The button to subtract a line of a QScrollBar.
::sub-page	The region between the handle (slider) and the sub-line of a QScrollBar.
::tab	The tab of a QTabBar or QToolBox.
::tab-bar	The tab bar of a QTabWidget. This subcontrol exists only to control the position of the QTabBar inside the QTabWidget. To style the tabs using the ::tab subcontrol.
::tear	The tear indicator of a QTabBar.
::tearoff	The tear-off indicator of a QMenu.
::text	The text of a QAbstractItemView.
::title	The title of a QGroupBox or a QDockWidget.
::up-arrow	The up arrow of a QHeaderView (sort indicator), QScrollBar or a QSpinBox.
::up-button	The up button of a QSpinBox.

See Customizing the QPushButton's Menu Indicator Sub-Control for an example of how to customize a subcontrol.

Customizing Qt Widgets Using Style Sheets Qt Style Sheets Examples

Hide Notes

Notes provided by the Qt Community

Sorted by:

Rating
Date

outline attribute

There is another property that is stylable, and that is not in this list of stylable properties above: outline.

This property can at least be used to remove the focus rectangle around buttons, if you use it like this:

```
1. //style sheet
2. QPushButton: {
3.     outline: none;
4. }
```

It might have more uses.

Informative
4

Votes: 4
Coverage: Qt library 4.7, 4.8, qt 5.0

Andre
Robot Herder
30 notes
Qt Certified Developer

[Revisions]