Qt基础控件美化

TamiGroup

Xin.Zhang

2017-02-22

Qt 美化之基础控件美化

前言

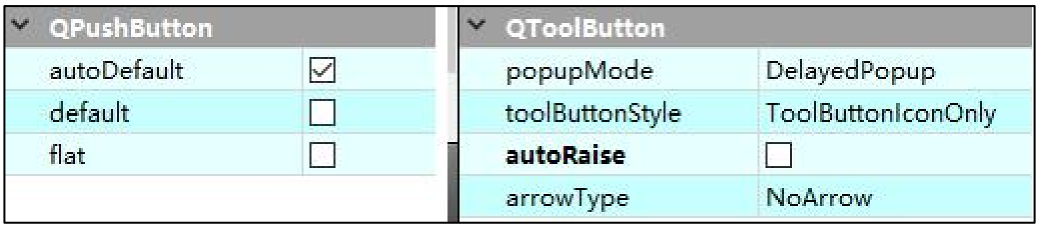
目标：只使用 CSS 的方式，展示每种控件的可美化内容和细节。

文中使用的 CSS 样式都是通过如下代码的方式加载到窗口中，文件以 UTF-8 进行存储。

```
QDir::setCurrent(QApplication::applicationDirPath());
QByteArray bCSS;
QFile fCSS("skin.css");
if( fCSS.open(QIODevice::ReadOnly) ){
bCSS = fCSS.readAll();
fCSS.close();
}
this->setStyleSheet(QString::fromUtf8(bCSS));
```

1. Button 篇

QPushButton 与 QToolButton 区别：



QPushButton	
autoDefault	☑
default	☐
flat	☐

QToolButton	
popupMode	DelayedPopup
toolButtonStyle	ToolButtonIconOnly
autoRaise	☐
arrowType	NoArrow

QToolButton 多与 QToolBar 一起使用，它有个关联 action。而 QPushButton 是 QpushButton:普通的下压式按钮，触发式事件信号不同。

1.1 QPushButton

它支持盒子模型，支持 :default, :flat, :checked 三种伪状态。

对于有菜单的按钮，菜单指示器可以使用 ::menu-indicator 子控件进行修饰。同时可以定制有菜单按钮的 :open（菜单打开） 和 :closed（菜单关闭） 状态。

注意：如果你使用 background-color 属性设置了颜色，一定要设置 border 属性，负责不起效。因为，绘制默认的按钮边框时，会覆盖背景颜色。

```
QPushButton {
border: 2px solid #8f8f91;
border-radius: 6px;
background-color: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #f6f7fa, stop: 1 #dadbde);
min-width: 80px;
}
QPushButton:pressed {
```

```
background-color: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #dadbde, stop: 1 #f6f7fa);
}
QPushButton:flat {
border: none; /* no border for a flat push button */
}
QPushButton:default {
border-color: navy; /* make the default button prominent */
  }
```

```
QPushButton:open { /* when the button has its menu open */
background-color: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #dadbde, stop: 1 #f6f7fa);
}
QPushButton::menu-indicator {
image: url(menu_indicator.png);
subcontrol-origin: padding;
subcontrol-position: bottom right;
}
QPushButton::menu-indicator:pressed, QPushButton::menu-indicator:open {
position: relative;
top: 2px; left: 2px; /* shift the arrow by 2 px */
}
```

```
QPushButton {   /*Border Image 的使用*/
color: grey;
border-image: url(/home/kamlie/code/button.png) 3 10 3 10;
border-top: 3px transparent;
border-bottom: 3px transparent;
border-right: 10px transparent;
border-left: 10px transparent;
}
```

1.2 QToolButton

它支持盒子模型。

对于有菜单的按钮，菜单指示器可以使用 ::menu-indicator 子控件进行修饰。默认的菜单指示器是在按钮的右下角。

如果处于 QToolButton::MenuButtonPopup 模式，::menu-button 子控件可以用来绘制菜单按钮，::menu-arrow 子控件用来绘制菜单箭头，默认菜单在右方居中位置。

当 QToolButton 展示箭头时，::up-arrow, ::down-arrow, ::left-arrow 和 ::right-arrow 子控件可以使用。

注意：如果你使用 background-color 属性设置了颜色，一定要设置 border 属性，负责不起效。因为，绘制默认的按钮边框时，会覆盖背景颜色。

QToolButton 没有菜单时，样式类同 QPushButton。

QToolButton 有菜单，并且设置 QToolButton::popupMode 为 QToolButton::DelayedPopup 或 QToolButton::InstantPopup 时，它与有菜单的 QPushButton 类同。

当处于 QToolButton::MenuButtonPopup 模式时，按照如下样式表：

```
QToolButton { /* all types of tool button */
border: 2px solid #8f8f91;
border-radius: 6px;
background-color: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #f6f7fa, stop: 1 #dadbde);
}
QToolButton[popupMode="1"] { /* only for MenuButtonPopup */
padding-right: 20px; /* make way for the popup button */
}
QToolButton:pressed {
background-color: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #dadbde, stop: 1 #f6f7fa);
}
/* the subcontrols below are used only in the MenuButtonPopup mode */
QToolButton::menu-button {
border: 2px solid gray;
border-top-right-radius: 6px;
border-bottom-right-radius: 6px;
/* 16px width + 4px for border = 20px allocated above */
width: 16px;
}
QToolButton::menu-arrow {
image: url(downarrow.png);
}
QToolButton::menu-arrow:open {
top: 1px; left: 1px; /* shift it a bit */
}
```

1.3 QRadioButton

它支持盒子模型。
选中状态指示器可用 ::indicator 子控件进行修饰。默认的菜单指示器是在控件的左上角。
样式中的 spacing 属性定义了指示器与文本的距离。spacing: 10。

```
QRadioButton::indicator {
width: 13px;
height: 13px;
}
QRadioButton::indicator::unchecked {
image: url(:/images/radiobutton_unchecked.png);
}
QRadioButton::indicator:unchecked:hover {
image: url(:/images/radiobutton_unchecked_hover.png);
}
QRadioButton::indicator:unchecked:pressed {
image: url(:/images/radiobutton_unchecked_pressed.png);
}
QRadioButton::indicator::checked {
image: url(:/images/radiobutton_checked.png);
}
QRadioButton::indicator:checked:hover {
image: url(:/images/radiobutton_checked_hover.png);
}
QRadioButton::indicator:checked:pressed {
image: url(:/images/radiobutton_checked_pressed.png);
}
```

1.4 QCheckBox

它支持盒子模型。
选中状态指示器可用 ::indicator 子控件进行修饰。默认的菜单指示器是在控件的左上角。
样式中的 spacing 属性定义了指示器与文本的距离。spacing: 10。
它与 QRadioButton 几乎完全相同，不同的是多了一个 :indeterminate 状态。

```
QCheckBox {
    spacing: 5px;
}

QCheckBox::indicator {
    width: 13px;
    height: 13px;
}

QCheckBox::indicator:unchecked {
    image: url(:/images/checkbox_unchecked.png);
}

QCheckBox::indicator:unchecked:hover {
    image: url(:/images/checkbox_unchecked_hover.png);
}

QCheckBox::indicator:unchecked:pressed {
    image: url(:/images/checkbox_unchecked_pressed.png);
}

QCheckBox::indicator:checked {
    image: url(:/images/checkbox_checked.png);
}

QCheckBox::indicator:checked:hover {
      image: url(:/images/checkbox_checked_hover.png);
}

QCheckBox::indicator:checked:pressed {
      image: url(:/images/checkbox_checked_pressed.png);
}

QCheckBox::indicator:indeterminate:hover {
      image: url(:/images/checkbox_indeterminate_hover.png);
}

QCheckBox::indicator:indeterminate:pressed {
      image: url(:/images/checkbox_indeterminate_pressed.png);
}
```

QCommandLinkButton 和 QDialogButtonBox 不做说明。

2. Item 篇

Qt 当中，分为 Model-based(Views) 和 Item-Based(Widgets)，但是他们的修饰方式是一样的。

2.1 QListWidget(等同 QListView)

它支持盒子模型。
当交替行颜色启用，可以通过 alternate-background-color 属性设置。
选中项目的颜色和背景颜色，可以通过 selection-color 和 selection-background-color 分别设置。
选中区域的行为可以通过 show-decoration-selected 属性设置。
使用::item 子控件可以设置每个 item 的具体样式。
参看 QAbsractScrollArea 去定制可滚动区域的背景样式。

```
QListView {
alternate-background-color: yellow; /*交替行颜色设置*/
}
```

```
QListView {
show-decoration-selected: 1; /* make the selection span the entire width of the view */
}
QListView::item:alternate {
background: #EEEEEE;
}
```

```
QListView::item:selected {
border: 1px solid #6a6ea9;
}
QListView::item:selected:!active {
background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #ABAFE5, stop: 1 #8588B2);
}
QListView::item:selected:active {
background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #6a6ea9, stop: 1 #888dd9);
}
QListView::item:hover {
background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #FAFBFE, stop: 1 #DCDEF1);
}
```

2.2 QTreeWidget(等同 QTreeView)

它支持盒子模型。
当交替行颜色启用，可以通过 alternate-background-color 属性设置。
选中项目的颜色和背景颜色，可以通过 selection-color 和 selection-background-color 分别设置。
选中区域的行为可以通过 show-decoration-selected 属性设置。
使用::item 子控件可以设置每个 item 的具体样式。
参看 QAbsractScrollArea 去定制可滚动区域的背景样式。
树视图的分支可以使用::branch 子控件进行定制，::branch 支持 :open, :closed, :has-sibling 和 :has-children 伪状态。

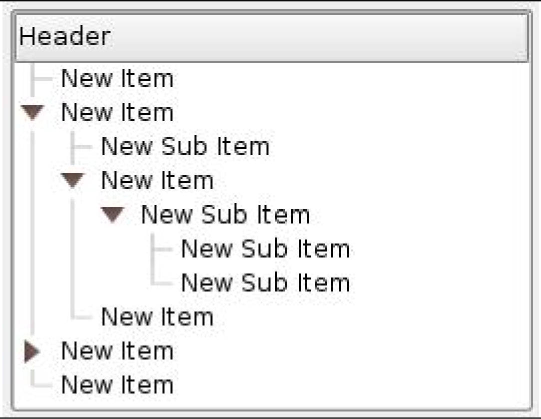
```
QTreeView {
alternate-background-color: yellow;
}
```

```
QTreeView {
show-decoration-selected: 1;
}
QTreeView::item {
border: 1px solid #d9d9d9;
border-top-color: transparent;
border-bottom-color: transparent;
}
QTreeView::item:hover {
background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1, stop: 0 #e7effd, stop: 1 #cbdaf1);
border: 1px solid #bfcde4;
}
QTreeView::item:selected {
```

```
border: 1px solid #567dbc;
}
QTreeView::item:selected:active{
background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1, stop: 0 #6ea1f1, stop: 1 #567dbc);
}
QTreeView::item:selected:!active {
background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1, stop: 0 #6b9be8, stop: 1 #577fbf);
}
```

```
QTreeView::branch {
background: palette(base);
}
QTreeView::branch:has-siblings:!adjoins-item {
background: cyan;
}
QTreeView::branch:has-siblings:adjoins-item {
background: red;
}
QTreeView::branch:!has-children:!has-siblings:adjoins-item {
background: blue;
}
QTreeView::branch:closed:has-children:has-siblings {
background: pink;
}
```

```
QTreeView::branch:has-siblings:!adjoins-item {
border-image: url(vline.png) 0;
}
QTreeView::branch:has-siblings:adjoins-item {
border-image: url(branch-more.png) 0;
}
QTreeView::branch:!has-children:!has-siblings:adjoins-item {
border-image: url(branch-end.png) 0;
}
QTreeView::branch:has-children:!has-siblings:closed,
QTreeView::branch:closed:has-children:has-siblings {
border-image: none;
image: url(branch-closed.png);
}
QTreeView::branch:open:has-children:!has-siblings,
QTreeView::branch:open:has-children:has-siblings    {
border-image: none;
image: url(branch-open.png);
}
```



2.3 QTableWidget(等同 QTableView)

它支持盒子模型。

当交替行颜色启用，可以通过 alternate-background-color 属性设置。

选中项目的颜色和背景颜色，可以通过 selection-color 和 selection-background-color 分别设置。

边角的按钮是通过 QAbstractButton 实现的，可用使用 QTableView QTableCornerButton::section 选择器进行修饰。

表格的颜色可以使用 gridline-color 属性进行设置。

参看 QAbsractScrollArea 去定制可滚动区域的背景样式。

注意：QTableCornerButton 的背景颜色与 Border 需要同时设置，原因同 Button。

```
QTableView {
selection-background-color: qlineargradient(x1: 0, y1: 0, x2: 0.5, y2: 0.5,
stop: 0 #FF92BB, stop: 1 white);
}
```

```
QTableView QTableCornerButton::section {
background: red;
border: 2px outset red;
}
```

3. Container 篇

3.1 QGroupBox

它支持盒子模型。

标题可以使用::title 子控件进行修饰。默认的标题位置依赖 QGroupBox::textAlignment 属性。

可选中的 QGroupBox 中，标题包含指示器，指示器可以使用 ::indicator 进行修饰。

样式属性 spacing 可以设置指示器与文本之间的距离。

```
QGroupBox {
```

```
background-color: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #E0E0E0, stop: 1 #FFFFFF);
border: 2px solid gray;
border-radius: 5px;
margin-top: 1ex; /* leave space at the top for the title */
}
QGroupBox::title {
subcontrol-origin: margin;
subcontrol-position: top center; /* position at the top center */
padding: 0 3px;
background-color: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #FF0ECE, stop: 1 #FFFFFF);
   }
```

```
QGroupBox::indicator {
width: 13px;
height: 13px;
}
QGroupBox::indicator:unchecked {
image: url(:/images/checkbox_unchecked.png);
}
/* proceed with styling just like QCheckBox */
```

3.2 QTabWidget

Tab 控件中的每个 Frame 可以使用 ::pane 子控件进行修饰。

左角和右角可以使用 ::left-corner 和 ::right-corner 进行设置。TabBar 的位置使用 ::tab-bar 子控件进行设置。

默认地，tab 子控件使用 QWindowsStyle，需要设置 QTabBar 的位置，可以设置::tab-bar 子控件修饰。

伪状态 :top, :left, :right, :bottom 定义了 tabs 的方向。

```
QTabWidget::pane { /* The tab widget frame */
border-top: 2px solid #C2C7CB;
}
QTabWidget::tab-bar {
left: 5px; /* move to the right by 5px */
}
/* Style the tab using the tab sub-control. Note that
it reads QTabBar _not_ QTabWidget */
QTabBar::tab {
background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #E1E1E1, stop: 0.4 #DDDDDD,
stop: 0.5 #D8D8D8, stop: 1.0 #D3D3D3);
border: 2px solid #C4C4C3;
border-bottom-color: #C2C7CB; /* same as the pane color */
border-top-left-radius: 4px;
border-top-right-radius: 4px;
min-width: 8ex;
padding: 2px;
}
QTabBar::tab:selected, QTabBar::tab:hover {
background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #fafafa, stop: 0.4 #f4f4f4,
stop: 0.5 #e7e7e7, stop: 1.0 #fafafa);
}
QTabBar::tab:selected {
border-color: #9B9B9B;
border-bottom-color: #C2C7CB; /* same as pane color */
}
QTabBar::tab:!selected {
margin-top: 2px; /* make non-selected tabs look smaller */
   }
```

```
QTabWidget::pane { /* The tab widget frame */
border-top: 2px solid #C2C7CB;
}
QTabWidget::tab-bar {
left: 5px; /* move to the right by 5px */
}
/* Style the tab using the tab sub-control. Note that
it reads QTabBar _not_ QTabWidget */
QTabBar::tab {
background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #E1E1E1, stop: 0.4 #DDDDDD,
stop: 0.5 #D8D8D8, stop: 1.0 #D3D3D3);
border: 2px solid #C4C4C3;
border-bottom-color: #C2C7CB; /* same as the pane color */
border-top-left-radius: 4px;
border-top-right-radius: 4px;
min-width: 8ex;
padding: 2px;
}
QTabBar::tab:selected, QTabBar::tab:hover {
background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #fafafa, stop: 0.4 #f4f4f4,
stop: 0.5 #e7e7e7, stop: 1.0 #fafafa);
}
QTabBar::tab:selected {
```

```
border-color: #9B9B9B;
border-bottom-color: #C2C7CB; /* same as pane color */
}
QTabBar::tab:!selected {
margin-top: 2px; /* make non-selected tabs look smaller */
}
/* make use of negative margins for overlapping tabs */
QTabBar::tab:selected {
/* expand/overlap to the left and right by 4px */
margin-left: -4px;
margin-right: -4px;
}
QTabBar::tab:first:selected {
margin-left: 0; /* the first selected tab has nothing to overlap with on the left */
}
QTabBar::tab:last:selected {
margin-right: 0; /* the last selected tab has nothing to overlap with on the right */
}
QTabBar::tab:only-one {
margin: 0; /* if there is only one tab, we don't want overlapping margins */
}
To move the tab bar to the center (as below), we require the following stylesheet:
￼
QTabWidget::pane { /* The tab widget frame */
border-top: 2px solid #C2C7CB;
position: absolute;
top: -0.5em;
}
QTabWidget::tab-bar {
alignment: center;
}
/* Style the tab using the tab sub-control. Note that
it reads QTabBar _not_ QTabWidget */
QTabBar::tab {
background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #E1E1E1, stop: 0.4 #DDDDDD,
stop: 0.5 #D8D8D8, stop: 1.0 #D3D3D3);
border: 2px solid #C4C4C3;
border-bottom-color: #C2C7CB; /* same as the pane color */
border-top-left-radius: 4px;
border-top-right-radius: 4px;
min-width: 8ex;
padding: 2px;
}
```

```
QTabBar::tab:selected, QTabBar::tab:hover {
background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #fafafa, stop: 0.4 #f4f4f4,
stop: 0.5 #e7e7e7, stop: 1.0 #fafafa);
}
QTabBar::tab:selected {
border-color: #9B9B9B;
border-bottom-color: #C2C7CB; /* same as pane color */
}
```

```
QTabBar::tear {
image: url(tear_indicator.png);
}
QTabBar::scroller { /* the width of the scroll buttons */
width: 20px;
}
QTabBar QToolButton { /* the scroll buttons are tool buttons */
border-image: url(scrollbutton.png) 2;
border-width: 2px;
}
QTabBar QToolButton::right-arrow { /* the arrow mark in the tool buttons */
image: url(rightarrow.png);
}
QTabBar QToolButton::left-arrow {
image: url(leftarrow.png);
   }
```

3.3 QTabBar

单独的 tabs 可以使用::tab 子控件修饰。关闭按钮使用 ::close-button 修饰。Tabs 支持 :only-one, :first, :last, :middle, :previous--selected, :next-selected, :selected 伪状态。

Tear 指示器(右边可以左右选择那个)使用 ::tear 子控件进行修饰。

QTabBar 使用两个 QToolButtons 进行 scrollers 控制，你可以使用 QTabBar QToolButton 选择器。::scroller 子控件可以同时设置他们的宽度。

QTabBar 的对齐方式使用 alignment 属性设置。

注意：设置 QTabWidget 当中的 QTabBar 的位置，需要通过 QTabWidget::tab-bar 子控件进行修改。

3.4 QToolBox

支持盒子模型。

单独的 tabs 可通过::tab 子控件进行设置，tabs 支持:only-one, :first, :last, :middle, :previous-selected, :next-selected, :selected 伪选择器。

3.5 QDockWidget

支持定制标题栏和标题栏按钮。

边框可以通过 border 属性定义，::title 子空间用来定制标题栏，::close-button 与::float-button 用来定制标题栏上的关闭按钮和浮动按钮。

当标题栏是垂直的，:vertical 伪状态被设置。同时依赖于 QDockWidget::DockWidgetFeature， :closable, :floatable 和:movable 伪状态被设置。

QMainWindow::separator 用来定制 resize handle。

当 QDockWidget 处于 undocked 状态时，样式表无效。

```
QDockWidget {
border: 1px solid lightgray;
titlebar-close-icon: url(close.png);
titlebar-normal-icon: url(undock.png);
}
QDockWidget::title {
text-align: left; /* align the text to the left */
background: lightgray;
padding-left: 5px;
}
QDockWidget::close-button, QDockWidget::float-button {
border: 1px solid transparent;
background: darkgray;
padding: 0px;
}
QDockWidget::close-button:hover, QDockWidget::float-button:hover {
background: gray;
}
QDockWidget::close-button:pressed, QDockWidget::float-button:pressed {
padding: 1px -1px -1px 1px;
  }
```

```
QDockWidget {
border: 1px solid lightgray;
titlebar-close-icon: url(close.png);
titlebar-normal-icon: url(float.png);
}
QDockWidget::title {
text-align: left;
background: lightgray;
padding-left: 35px;
}
QDockWidget::close-button, QDockWidget::float-button {
background: darkgray;
padding: 0px;
icon-size: 14px; /* maximum icon size */
}
QDockWidget::close-button:hover, QDockWidget::float-button:hover {
background: gray;
}
QDockWidget::close-button:pressed, QDockWidget::float-button:pressed {
padding: 1px -1px -1px 1px;
}
QDockWidget::close-button {
subcontrol-position: top left;
subcontrol-origin: margin;
position: absolute;
top: 0px; left: 0px; bottom: 0px;
width: 14px;
}
QDockWidget::float-button {
subcontrol-position: top left;
subcontrol-origin: margin;
position: absolute;
top: 0px; left: 16px; bottom: 0px;
width: 14px;
   }
```

4. InputWidget 篇

4.1 QLineEdit

它支持盒子模型。

选中内容的颜色和背景颜色可以通过 selection-color 和 selection-background-color 来设置。

密码类型的字符可以通过 lineedit-password-character 属性设置，密码字符的延时通过 lineedit-password-mask-delay 来设置。

```
QLineEdit {
border: 2px solid gray;
border-radius: 10px;
padding: 0 8px;
background: yellow;
selection-background-color: darkgray;
}
```

```
QLineEdit[echoMode="2"] {
lineedit-password-character: 9679;
}
QLineEdit:read-only {
background: lightblue;
}
```

4.2 QTextEdit

它支持盒子模型。
选中内容的颜色和背景颜色可以通过 selection-color 和 selection-background-color 来设置。
可滚动区域的背景颜色参照 QAbsractScrollArea 。

4.3 QComboBox

它支持盒子模型。
下拉按钮可以通过::drop-down 子控件设置，默认在右上角。下拉按钮的图标可以通过::down-arrow 子控件设置。
QComboBox 的弹出部分是 QAbstractItemView 控件，可以使用选择器设置。

```
QComboBox {
border: 1px solid gray;
border-radius: 3px;
padding: 1px 18px 1px 3px;
min-width: 6em;
}
QComboBox:editable {
background: white;
}
QComboBox:!editable, QComboBox::drop-down:editable {
background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #E1E1E1, stop: 0.4 #DDDDDD,
stop: 0.5 #D8D8D8, stop: 1.0 #D3D3D3);
}
/* QComboBox gets the "on" state when the popup is open */
QComboBox:!editable:on, QComboBox::drop-down:editable:on {
background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
stop: 0 #D3D3D3, stop: 0.4 #D8D8D8,
stop: 0.5 #DDDDDD, stop: 1.0 #E1E1E1);
}
QComboBox:on { /* shift the text when the popup opens */
padding-top: 3px;
padding-left: 4px;
}
QComboBox::drop-down {
```

```
subcontrol-origin: padding;
subcontrol-position: top right;
width: 15px;
border-left-width: 1px;
border-left-color: darkgray;
border-left-style: solid; /* just a single line */
border-top-right-radius: 3px; /* same radius as the QComboBox */
border-bottom-right-radius: 3px;
}
QComboBox::down-arrow {
image: url(/usr/share/icons/crystalsvg/16x16/actions/1downarrow.png);
}
QComboBox::down-arrow:on { /* shift the arrow when popup is open */
top: 1px;
left: 1px;
   }
```

```
QComboBox QAbstractItemView {
border: 2px solid darkgray;
selection-background-color: lightgray;
   }
```

4.4 QSpinBox（同 QTimeEdit,QDateTimeEdit）

它支持盒子模型。
向上的按钮和图标可以通过::up-button 和::up-arrow 子控件来设置，默认按钮占据控件高度的一半。
向下按钮同样可以使用::down-button 和::down-arrow 来设置。

```
QSpinBox {
padding-right: 15px; /* make room for the arrows */
border-image: url(:/images/frame.png) 4;
border-width: 3;
}
QSpinBox::up-button {
subcontrol-origin: border;
subcontrol-position: top right; /* position at the top right corner */
width: 16px; /* 16 + 2*1px border-width = 15px padding + 3px parent border */
border-image: url(:/images/spinup.png) 1;
border-width: 1px;
}
QSpinBox::up-button:hover {
border-image: url(:/images/spinup_hover.png) 1;
}
QSpinBox::up-button:pressed {
```

```
border-image: url(:/images/spinup_pressed.png) 1;
}
QSpinBox::up-arrow {
image: url(:/images/up_arrow.png);
width: 7px;
height: 7px;
}
QSpinBox::up-arrow:disabled, QSpinBox::up-arrow:off { /* off state when value is max */
image: url(:/images/up_arrow_disabled.png);
}
QSpinBox::down-button {
subcontrol-origin: border;
subcontrol-position: bottom right; /* position at bottom right corner */
width: 16px;
border-image: url(:/images/spindown.png) 1;
border-width: 1px;
border-top-width: 0;
}
QSpinBox::down-button:hover {
border-image: url(:/images/spindown_hover.png) 1;
}
QSpinBox::down-button:pressed {
border-image: url(:/images/spindown_pressed.png) 1;
}
QSpinBox::down-arrow {
image: url(:/images/down_arrow.png);
width: 7px;
height: 7px;
}
QSpinBox::down-arrow:disabled,
QSpinBox::down-arrow:off { /* off state when value in min */
image: url(:/images/down_arrow_disabled.png);
}
```

4.5 QScrollBar（包含水平和垂直）

它支持盒子模型，它的矩形区域包含外面的刻度。水平或者垂直，使用:horizontal 和:vertical 来表示。

滑块使用::handle 子控件控制，min-width 和 min-height 提供滑块的约束。

Add-line 按钮使用::add-line 子控件修饰，默认地水平右边(图标::right-arrow)或者垂直下边(图标::down-arrow)。

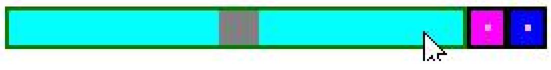
Sub-line 按钮使用::sub-line 子控件修饰,默认水平左边(图标::left-arrow)或者垂直上边(图标::up-arrow)。

Sub-page 区域使用::sub-page 子控件修饰，Add-page 区域使用::add-page 子控件修饰。

```
QScrollBar:horizontal {
border: 2px solid grey;
background: #32CC99;
height: 15px;
margin: 0px 20px 0 20px;
}
QScrollBar::handle:horizontal {
background: white;
min-width: 20px;
}
QScrollBar::add-line:horizontal {
border: 2px solid grey;
background: #32CC99;
width: 20px;
subcontrol-position: right;
subcontrol-origin: margin;
}
QScrollBar::sub-line:horizontal {
border: 2px solid grey;
background: #32CC99;
width: 20px;
subcontrol-position: left;
subcontrol-origin: margin;
    }
```

```
QScrollBar:left-arrow:horizontal, QScrollBar::right-arrow:horizontal {
border: 2px solid grey;
width: 3px;
height: 3px;
background: white;
}
QScrollBar::add-page:horizontal, QScrollBar::sub-page:horizontal {
background: none;
    }
```

```
QScrollBar:horizontal {
border: 2px solid green;
background: cyan;
height: 15px;
margin: 0px 40px 0 0px;
}
QScrollBar::handle:horizontal {
background: gray;
min-width: 20px;
}
QScrollBar::add-line:horizontal {
background: blue;
width: 16px;
subcontrol-position: right;
subcontrol-origin: margin;
border: 2px solid black;
}
QScrollBar::sub-line:horizontal {
background: magenta;
width: 16px;
subcontrol-position: top right;
subcontrol-origin: margin;
border: 2px solid black;
position: absolute;
right: 20px;
}
QScrollBar:left-arrow:horizontal, QScrollBar::right-arrow:horizontal {
width: 3px;
height: 3px;
background: pink;
}
QScrollBar::add-page:horizontal, QScrollBar::sub-page:horizontal {
background: none;
}
```



4.6 QSlider（包含水平和垂直）

它支持盒子模型。

对于水平滑块，min-width 和 height 必须提供。对于垂直滑块，min-height 和 width 必须提供。滑块沟槽使用::groove 修饰。滑块使用::handle 子控件修饰。

```
    QSlider::groove:horizontal {
    border: 1px solid #999999;
    height: 8px; /* the groove expands to the size of the slider by default. by giving it a height, it has a fixed size
*/
    background: qlineargradient(x1:0, y1:0, x2:0, y2:1, stop:0 #B1B1B1, stop:1 #c4c4c4);
    margin: 2px 0;
    }
    QSlider::handle:horizontal {
    background: qlineargradient(x1:0, y1:0, x2:1, y2:1, stop:0 #b4b4b4, stop:1 #8f8f8f);
    border: 1px solid #5c5c5c;
    width: 18px;
    margin: -2px 0; /* handle is placed by default on the contents rect of the groove. Expand outside the groove
*/
    border-radius: 3px;
    }
```

```
    QSlider::groove:vertical {
    background: red;
    position: absolute; /* absolutely position 4px from the left and right of the widget. setting margins on the
widget should work too... */
    left: 4px; right: 4px;
    }
    QSlider::handle:vertical {
    height: 10px;
    background: green;
    margin: 0 -4px; /* expand outside the groove */
    }
    QSlider::add-page:vertical {
    background: white;
    }
    QSlider::sub-page:vertical {
    background: pink;
      }
```

5. DisplayWidget 篇

5.1 QLabel（类似 QFrame）

它支持盒子模型,不支持:hover。
对 QLabel 设置样式表会自动将 QFrame::frameStyle 属性赋值给 QFrame::StyledPanel。

```
    QFrame, QLabel, QToolTip {
    border: 2px solid green;
    border-radius: 4px;
    padding: 2px;
    background-image: url(images/welcome.png);
      }
```

5.2 QProgressBar

它支持盒子模型。

显示的块使用::chunk 子控件修饰。
如果需要展示进度文本，使用 text-align 进行文本位置定义。
不定状态的进度条，可使用:indeterminate 伪状态进行定义。

```
QProgressBar {
border: 2px solid grey;
border-radius: 5px;
text-align: center;
}
QProgressBar::chunk {
background-color: #05B8CC;
width: 20px;
   }
```

5.4 其他展示控件

- LCD Number
- Graphics View
- OpenGL Widget
- QQuickWidget
- QToolTip

6. 其他重要控件

6.1 QMenu

它支持盒子模型。
单独的项可通过::item 子控件进行修饰，它支持 :selected, :default, :exclusive 和 the non-exclusive 四种伪状态。
可选中 menu 项的指示器可通过::indicator 子控件进行修饰。
有子菜单的项指示图标使用:right-arrow 和:left-arrow 进行修饰。
向上选择的图标使用::scroller，向下选择的图标使用::tearoff。

QMenuBar: QMenu 的容器。

```
QMenu {
background-color: #ABABAB; /* sets background of the menu */
border: 1px solid black;
}
QMenu::item {
```

```
/* sets background of menu item. set this to something non-transparent
if you want menu color and menu item color to be different */
background-color: transparent;
}
QMenu::item:selected { /* when user selects item using mouse or keyboard */
background-color: #654321;
  }
```

```
QMenu {
background-color: white;
margin: 2px; /* some spacing around the menu */
}
QMenu::item {
padding: 2px 25px 2px 20px;
border: 1px solid transparent; /* reserve space for selection border */
}
QMenu::item:selected {
border-color: darkblue;
background: rgba(100, 100, 100, 150);
}
QMenu::icon:checked { /* appearance of a 'checked' icon */
background: gray;
border: 1px inset gray;
position: absolute;
top: 1px;
right: 1px;
bottom: 1px;
left: 1px;
}
QMenu::separator {
height: 2px;
background: lightblue;
margin-left: 10px;
margin-right: 5px;
}
QMenu::indicator {
width: 13px;
height: 13px;
}
/* non-exclusive indicator = check box style indicator (see QActionGroup::setExclusive) */
QMenu::indicator:non-exclusive:unchecked {
image: url(:/images/checkbox_unchecked.png);
}
QMenu::indicator:non-exclusive:unchecked:selected {
image: url(:/images/checkbox_unchecked_hover.png);
}
QMenu::indicator:non-exclusive:checked {
image: url(:/images/checkbox_checked.png);
}
QMenu::indicator:non-exclusive:checked:selected {
image: url(:/images/checkbox_checked_hover.png);
}
/* exclusive indicator = radio button style indicator (see QActionGroup::setExclusive) */
QMenu::indicator:exclusive:unchecked {
image: url(:/images/radiobutton_unchecked.png);
}
QMenu::indicator:exclusive:unchecked:selected {
image: url(:/images/radiobutton_unchecked_hover.png);
}
QMenu::indicator:exclusive:checked {
image: url(:/images/radiobutton_checked.png);
}
QMenu::indicator:exclusive:checked:selected {
image: url(:/images/radiobutton_checked_hover.png);
   }
```

6.2 QHeaderView

它支持盒子模型。

它的选区可以使用::section 子控件修饰，它支持:middle, :first, :last, :only-one, :next-selected, :previous-selected, :selected, 和 :checked 几种伪状态。

排序指示器可以使用::up-arrow 和::down-arrow 子控件修饰。

```
QHeaderView::section {
background-color: qlineargradient(x1:0, y1:0, x2:0, y2:1,
stop:0 #616161, stop: 0.5 #505050,
stop: 0.6 #434343, stop:1 #656565);
color: white;
padding-left: 4px;
border: 1px solid #6c6c6c;
}
QHeaderView::section:checked
{
background-color: red;
}
/* style the sort indicator */
QHeaderView::down-arrow {
image: url(down_arrow.png);
}
```

```
    QHeaderView::up-arrow {
    image: url(up_arrow.png);
    }
```

6.3 QAbstractScrollArea

它支持盒子模型。

它的任何子类都支持可滚动的背景，通过 background-attachment 属性设置。background-image 用来设置图像。

```
QTextEdit, QListView {
background-color: white;
background-image: url(draft.png);
background-attachment: scroll;
}
```

```
QTextEdit, QListView {
background-color: white;
background-image: url(draft.png);
background-attachment: fixed;
  }
```

6.4 属性列表

有(*)星号标志的说明与 CSS 标准不一致。

属性	类型	描述

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 certair 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 }
lineedit-password-mask-dela	Number	The QLineEdit password mask delay in milliseconds before

y*		lineedit-password-character is applied to visible character. If this property is not specified, it defaults to the value specified by the current style for the SH_LineEdit_PasswordMaskDelay style hint. This property was added in Qt 5.4. Example: * { lineedit-password-mask-delay: 1000 }
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 }
outline		The outline drawn around the object's border.
outline-color	Color	The color of the outline. See also border-color
outline-offset	Length	The outline's offset from the border of the widget.
outline-style		Specifies the pattern used to draw the outline. See also border-style
outline-radius		Adds rounded corners to the outline
outline-bottom-left-radius	Radius	The radius for the bottom-left rounded corner of the outline.
outline-bottom-right-radius	Radius	The radius for the bottom-right rounded corner of the outline.
outline-top-left-radius	Radius	The radius for the top-left corner of the outline.
outline-top-right-radius	Radius	The radius for the top-right rounded corner of the outline.
padding	Box Lengths	The widget's padding. Equivalent to specifying padding-top, 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.

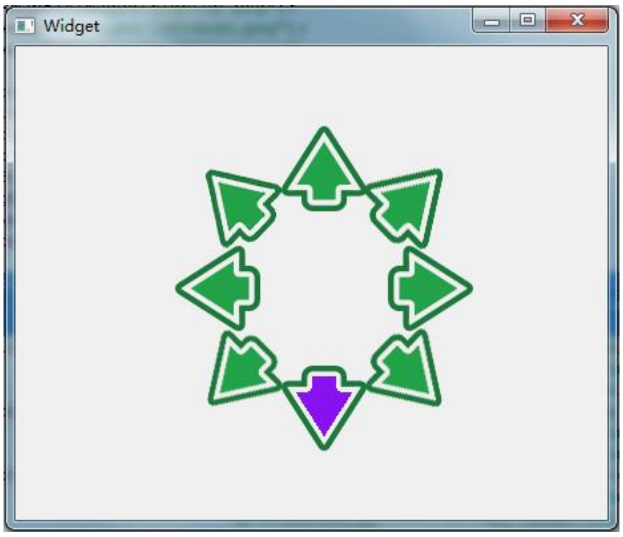
6.5 伪状态列表

伪状态	描述
: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)

7. 异型控件

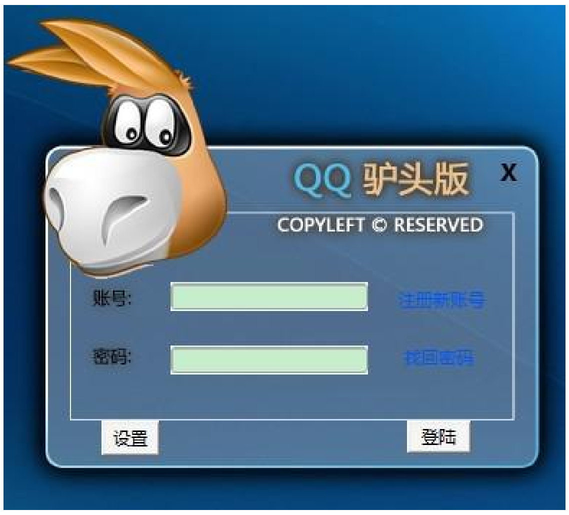
7.1 不规则图标按钮



实现方法：可以取掩码

```
QPixmap pixmap("images/left.png");//这里是一个左箭头
ui->leftButton->setMask(pixmap.mask());//setMask()函数来设置控件掩码，参数是从 pixmap 中提取的位图掩码 pixmap.m
```

7.2 异形窗口



实现方法; 在 main 函数中，加入重要代码

```
w.setWindowOpacity(1);//设置透明度，默认是 1 为全不透明，0 为全透明
w.setWindowFlags(Qt::FramelessWindowHint);//设置窗口为无边界
w.setAttribute(Qt::WA_TranslucentBackground);//设置窗口属性为背景半透明
```

7.3 设置背景图片

1. 通过 StyleSheet 设置。
2. 代码：

```
        QWidget *widget = new QWidget();
        widget->setAutoFillBackground(true);
        QPalette palette;
        QPixmap pixmap(":/Resources/Penguins.jpg");
        palette.setBrush(QPalette::Window, QBrush(pixmap));
        widget->setPalette(palette);
        widget->show();
```