1. 样式表的内容

QPushButton,QCheckBox::indicator,QLineEdit:hover

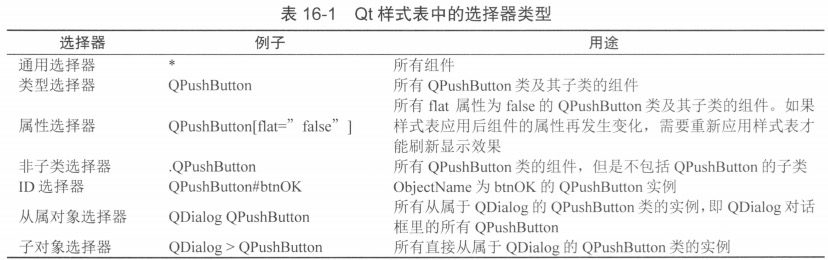
{

color : red;

font: 12pt “仿宋”;

}

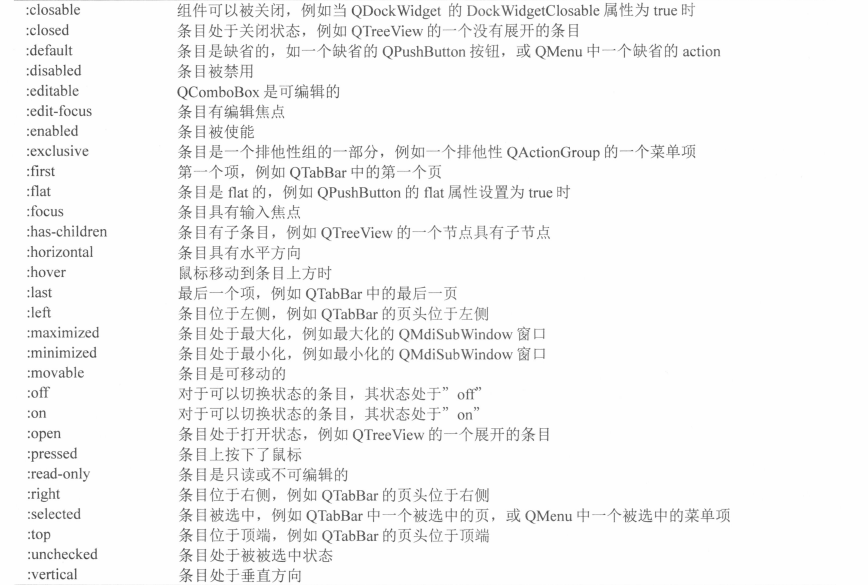
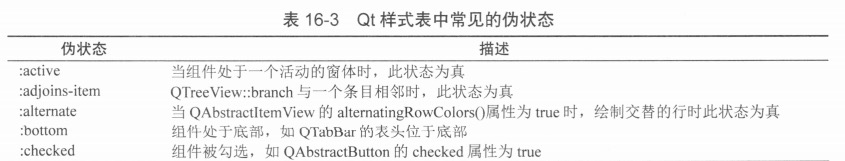
1. “QPushButton”“QCheckBox”“QLineEdit”: 选择器
2. “indicator”: 子控件,跟随在选择器的后面, 用双冒号隔开
3. “hover”: 伪状态, ,跟随在选择器的后面,或者选择器子控件的后面, 用单冒号隔开, 前面加一个”!”可以对状态取反
4. “color ”“font”: 样式的属性
5. “red”“12pt”“仿宋”: 样式的值, 跟随在属性后面,用单冒号隔开, 多个值之间用空格隔开
6. 常见的选择器类型



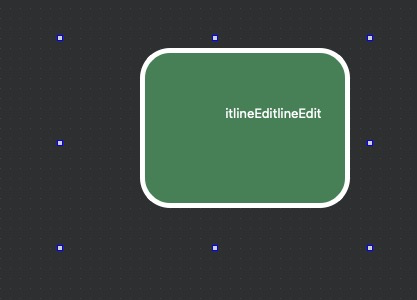
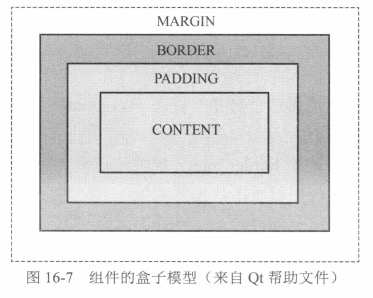
1. 常见的子控件列表



1. 常见的伪状态列表



1. 常见的属性



1. CONTENT表示的是内容区域, 对应第二张图的文字部分
2. BORDER表示的边框,对应第二张图的白色框
3. PADDING表示的是边框和内容之间的区域
4. MARGIN表示的是控件和其他控件之间的空白边距

max-width: 100px;

max-height: 100px;

min-width: 100px;

min-height: 100px;

padding: 10px 20px 40px 80px;

border-width: 5px;

border-style:solid;

border-color:white;

border-radius:30px;

margin: 10px 20px 40px 80px;

1. 一些场景中,有多个多种类型的控件为必填项,可以通过以下方式单独设置这些必填项的样式

首先给必填项的控件设置属性 :

editName->setProperty(“required”,”true”);

comboSex->setProperty(“required”,”true”);

checkAgree->setProperty(“required”,”true”);

然后给所有这些属性的控件设置样式:

\*[required="true"] {background-color: red}

1. 示例

//子控件的样式设置

QCheckBox::indicator

{

image: url(:/icon/icon/min.png);

}

//伪状态的样式设置

QCheckBox::hover

{

image: url(:/icon/icon/min.png);

}

//伪状态的串联设置,checkbox在选中且滑过状态下才显示黑色

QCheckBox:checked:hover

{

background-color: black;

}

//伪状态的并联设置,checkbox在选中或滑过状态下都显示黑色

QRadioButton:checked,QRadioButton:hover

{

background-color: black;

}

//子控件的伪状态样式设置

QCheckBox::indicator:hover

{

image: url(:/icon/icon/min.png);

}

1. 样式表的应用
2. 在ui界面内直接使用
3. 在代码中用setStyleSheet使用 ,

qApp->setStyleSheet();可以为应用程序设置全局样式;设置的内容可以从qss文件中读取

objectName->setStyleSheet();可以为一个界面组件单独设置样式;

1. 样式定义的明确性
2. 当多条同一级的样式冲突时,更明确的样式将被应用

QPushButton{color: red;}

QPushButton#button1 {color: blue;}

明确性: 对象>类, 所以button1为blue, 其他的按钮为red

QPushButton{color: red;}

QPushButton:hover {color: blue;}

明确性: 伪状态>非伪状态 所以当鼠标滑过按钮时为blue,其他状态为red

QPushButton:hover{color: red;}

QPushButton:enabled{color: blue;}

明确性: 相同 所以最后面的样式生效,在使能时为蓝色,不使能时为默认颜色

如果想要达到效果, 应该:

QPushButton:hover:enabled{color: red;} //在使能且鼠标滑过时为红色

QPushButton:enabled{color: blue;}//在使能且无鼠标滑过时为蓝色

QWidget{background-color: black;}

QPushButton{background-color: red;}

明确性: 相同 父子关系的两个类作为选择器,明确性相同, 所以最后面的样式生效,此时按钮为red, 窗体为black

QPushButton{background-color: red;}

QWidget{background-color: black;}

此时按钮为black, 窗体为black

1. 尽量避免样式冲突
2. 样式的级联性
3. 样式可以再qApp, 父窗口, 具体控件中设置, 任何一个组件的样式是以上各个级别样式的融合,当不同级别的样式发生冲突时, 当前组件会使用离自己最近的样式,而不用考虑样式的明确性

例:

qApp->setStyleSheet("QPushButton{color: red}");

parentwindow->setStyleSheet("QPushButton{color: blue}");

button->setStyleSheet("QPushButton{color: green}"); //父窗口是parentwindow

以上代码关于QPushButton的样式设置发生了冲突

在应用程序中: button按钮为green;

parentwindow窗口中除button外其他按钮为blue;

qApp其他窗口的按钮为red

1. 级别优先级>明确性优先级

qApp->setStyleSheet("QPushButton{color: red}");

parentwindow->setStyleSheet("QPushButton#pushButton{color: black}"

"QPushButton{color: blue}");

button->setStyleSheet("QPushButton{color: green}");

在应用程序中: button按钮为green;

parentwindow窗口中除button外其他按钮为blue;

qApp其他窗口的按钮为red

1. 样式设置无效(未解决)

顶级(没有父窗口)的Widget(widget\*)窗口,如果在ui或者代码中调用过widget\* 的stylesheet, 那么在widget\* 调用show之前, 构造函数中的widget->stylesheet()都是无效的

https://blog.csdn.net/qq\_24038299/article/details/117398450