借用 css 的灵感， Qt 也支持 Qt 自己的 css, 简称 qss。同 css 相似，qss 的主要功能与最目的都是能使界面的表现与界面的元素分离，即质与形的分离，就如同一个人可以在不同的时候穿上不同的衣服一样，css 机制的引入，使得设计一种皮肤与界面控件分离的软件成为可能，应用程序也能像 web 界面那样随意地改变外观。

1， QSS 语法

同 css 一样，他也有由一个 selector 与一个 declaration 组成，selector 指定了是对哪一个控件产生效果，而 declaration 才是真正的产生作用的语句。如：

```
QPushButton { color: red }
```

QPushButton 指定了是对所有的 QPushButton 或是其子类控件（如用户定义的 MyPushButton）产生影响，而 color:red 表明所有的受影响控件的前景色都为 red。

除了“类名”，“对象名”，“Qt 属性名”这三样东西是大小写敏感的外其他的东西都是大小写不敏感的，如 color 与 Color 代表同一属性。

如果有几个 selector 指定了相同的 declaration, 可以使用逗号“,”将各个选择器分开，如：

```
QPushButton, QLineEdit, QComboBox { color: red }
```

他相当于：

```
QPushButton { color: red }
QLineEdit { color: red }
QComboBox { color: red }
```

declaration 部份是一系列的（属性：值）对，使用分号“;”将各个不同的属性值对分开，使用大括号“{}”将所有 declaration 包含在一起。

1， 一般选择器（selector）

Qt 支持所有的 CSS2 定义的选择器，其祥细内容可以在 w3c 的网站上查找 http://www.w3.org/TR/CSS2/selector.html ， 其中比较常用的 selector 类型有：

i. 通用类型选择器：*会对所有控件有效果。

ii. 类型选择器：QPushButton 匹配所有 QPushButton 的实例和其子类的实例。

iii. 属性选择器：QPushButton[flat=”false”]匹配所有 QPushButton 属性 flat 为 false 的实例，属性分为两种，静态的和动态的，静态属性可以通过 Q_PROPERTY() 来指定，来动态属性可以使用 setProperty 来指定，如：

```
QLineEdit *nameEdit = new QLineEdit(this);
nameEdit->setProperty("mandatoryField", true);
```

如果在设置了 qss 后 Qt 属性改变了，需要重新设置 qss 来使其生效，可以使用先 unset 再 set qss。

iv. 类选择器：.QPushButton

匹配所有 QPushButton 的实例，但不包含其子类，这相当于：

```
*[class~="QPushButton"]
```

~=的意思是测试一个 QStringList 类型的属性是否包含给定的 QString。

v. ID 选择器：QPushButton#okButton

对应 Qt 里面的 object name 设置，使用这条 CSS 之前要先设置对应控件的 object name 为 okButton，如：

```
Ok->setObjectName(tr("okButton"));
```

vi. 后裔选择器：QDialog QPushButton

匹配所有为 QDialog 后裔（包含儿子，与儿子的儿子的递归）为 QPushButton 的实例

vii. 子选择器：QDialog > QPushButton

匹配所有的 QDialog 直接子类 QPushButton 的实例，不包含儿子的儿子的递归。

2， 子控件选择器

i. 对于复杂的控件，可能会在其中包含其他子控件，如一个 QComboxBox 中有一个 drop-down 的按钮。那么现在如果要设置 QComboxBox 的下拉按钮的话，就可以这样访问：

```
QComboBox::drop-down { image: url(dropdown.png) }
```

其标志是“::”

ii. 子控件选择器是用位置的引用来代表一个元素，这个元素可是一个单一控件或是另一个包含子控件的复合控件。使用 subcontrol-origin 属性可以改变子控件的默认放置位置，如：

```
QComboBox {
      margin-right: 20px;
}
QComboBox::drop-down {
      subcontrol-origin: margin;
}
```

如上语句可以用来改变 drop-down 的 margin。

iii. 相对位置属性可以用来改变子控件相对于最初位置的偏移量，如当一个 QCombox 的 drop-down 按钮被按下时，我们可以用一个内部的小偏移量来表示被按下的效果，如下：

```
QComboBox::down-arrow {
      image: url(down_arrow.png);
}
QComboBox::down-arrow:pressed {
      position: relative;
      top: 1px; left: 1px;
}
```

iv. 绝对位置属性允许子控件改变自己的位置和大小而不受引用元素的控件。一但位置被设定了，这些子控件就可以被当成一般的 widget 来对待，并且可以使用合模型。关于合模型可以参考下面。

3， 伪选择器（pseudo-states）

i. 伪选择器以冒号（:）表示，与 css 里的伪选择器相似，是基于控的一些基本状态来限定程序的规则，如 hover 规则表示鼠标经过控件时的状态，而 press 表示按下按钮时的状态。如：

```
QPushButton:hover {
    Background-color:red;
}
```

表示鼠标经过时 QPushButton 背景变红。

ii. Pseudo 还支持否定符号（！），如：

```
QRadioButton:!hover { color: red }
```

表示没有鼠标移上 QRadioButton 时他显示的颜色是 red。

iii. Pseudo 可以被串连在一起，比如：

```
QPushButton:hover:!pressed { color: blue; }
```

表示 QPushButton 在鼠标移上却没有点击时显示 blue 字，但如果点击的时候就不会显示 blue 颜色了。

iv. 同样可以和之前所讲的子控件选择器一起联合使用，如：

```
QSpinBox::down-button:hover
{ image: url(btn-combobox-press.bmp) }
```

v. 与前面所讲的一样，伪选择器，子控件选择器等都是可以用逗号“,”分隔表示连续相同的设置的，如：

```
QPushButton:hover,QSpinBox::down-button,
QCheckBox:checked
{ color: white ;image: url(btn-combobox-press.bmp);}
```

表示如下

更多请参考：

http://pepper.troll.no/s60prereleases/doc/stylesheet-reference.html#list-of-pseudo-states

2， 解决冲突

a) 使用 object name

i. 在程序里面要先设置控件的，如：

```
btnOne = new QPushButton(centralWidget);
btnOne->setObjectName(QString::fromUtf8("btnOneCh"));
```

ii. 这样，我们有了一个 object name 为 btnOneCh 的 QPushButton，试验一下，使用指定 object name 的方式，如：

```
QPushButton#btnOneCh { color: red }
QPushButton { color: white }
```

可以看出，btnOncCh 的 color 变成了 red

b) 使用伪选择器，如 hover，press，enabled 等，如：

按扭“1”是 disable 了的，

```
QPushButton:!enabled {color: white}
```

c) 所有的类型选择器都有一个共同的特性，就是如果有两个属性冲突了的话就会以最后出现的一个为准，如：

```
QPushButton { color: red }
QAbstractButton { color: gray }
```

由于 QPushButton 为 QAbstractButton 的子类，如果只设置 QAbstractButton 的可以想像结果是所有的 QPushButton 都为 gray, 如果只设置 QPushButton 的所有 QPushButton 都会为 red，当两个都能设置起效的时候，以在文本上最后出现的为准，所以结果为 Grey

d) 当然其中如果有#指定了 object name，他所设置的优先级是最大的，具体规则可以参考：
http://www.w3.org/TR/CSS2/cascade.html#specificity，或是
http://pepper.troll.no/s60prereleases/doc/stylesheet-syntax.html#conflict-resolution

```
QPushButton#btnOneCh { color: red }
QPushButton { color: blue }
QAbstractButton { color: gray }
```

虽然 QAbstractButton 在最后，但是之前有#btnOneCh 指定了 QPushButton “一”的 color 为 red 所以最后显示也是“一”为 red。

3， 级联效应

a) 子类可以继承父类的 StyleSheet，但是如果子类里面设置了 StyleSheet 与父类里在设置的有冲突，那么当然会优先考虑子类自己的，同样，如果在 qApp 时面设置了，但是在某一个特定控件里面也设置，如果有冲突，也是优先控件自己的，例如，我在程序时面设置了：

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

而，当我再设置 qApp 时，如果，将 QPushButton 的 color 设置成 grey 的，那么结果是对于 btnOneEn 这个 QPushButton 来说他的颜色还是 red。

这就是为什么这里设置为 grey 了 btnOneEn 却还是 red 的。

b) 如果我们对一个控件设置 StyleSheet 为：

```
QPushButton* myPushButton;
myPushButton->setStyleSheet("* { color: blue }");
```

其实他和设置为：

```
myPushButton->setStyleSheet("color: blue");
```

效果相同，只是后一种设置不会对 QPushButton 的子类产生作用，但第一种却会。

4， 继承性

与 CSS 不同的一点，在 CSS box 模型中，如果一个元素在别一元素的里面，那么里面的元素会自动继承外面元素的属性，但 QSS 里面不会，如：一个 QPushButton 如果放在一个 QGroupBox 里面，如果：

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

并不代表在 QGroupBox 里面的 QPushButton 也会有 color:red 的属性，如果要想有的话要显示写明，如：

```
qApp->setStyleSheet("QGroupBox, QGroupBox * { color: red; }");
```

或者在应用程序里面也可以用 QWidget::setFont 等来设置到子控件的属性。

5， Namespace 冲突

类型选择器能够使用到一个特定的类型，如：

```
class MyPushButton : public QPushButton {
// ...
}
qApp->setStyleSheet("MyPushButton { background: yellow; }");
```

因为 QSS 使用 QObject::className 来判断要赋与 style sheet 的控件类型，如果一个用户定义控件类型在一个 namespace 里面的话，QObject::className 会返回<namespace>::<classname> 的名字，这和子控件选择器的语法相冲突，为了解决此问题，使用“--”来代替“::”，比如：

```
namespace ns {
      class MyPushButton : public QPushButton {
          // ...
      }
 }
qApp->setSytleSheet("ns--MyPushButton { background:
yellow; }");
```

6， 设置对像属性

如果在程序里面使用 Q_PROPERTY 设置的属性，可以在 qss 里面使用：qproperty-<property name>的形式来访问并设置值。如：

```
MyLabel { qproperty-pixmap: url(pixmap.png); }
MyGroupBox { qproperty-titleColor: rgb(100, 200, 100); }
QPushButton { qproperty-iconSize: 20px 20px; }
```

如果属性引用到的是一个由 Q_ENUMS 申明的 enum 时，要引用其属性名字要用定义的名称而不是数字。

引用：

http://pepper.troll.no/s60prereleases/doc/stylesheet