在《[Qt Quick 简介](http://blog.csdn.net/foruok/article/details/28620525" \t "_blank)》中我们提到 QML 语法和 Json 类似，请参考《[Qt on Android: http下载与Json解析](http://blog.csdn.net/foruok/article/details/27556135" \t "_blank)》查看 Json 语法。当然这里我们是期望从零开始也能学会 QML ，所以呢，你也可以直接往下看。

    请给我的决赛文章《[Qt Quick 图像处理实例之美图秀秀（附源码下载）](http://vote.blog.csdn.net/Article/Details?articleid=37740583)》投票，谢谢。

    版权所有 foruok，转载请注明出处：[http://blog.csdn.net/foruok](http://blog.csdn.net/foruok" \t "_blank)

**对象**

   QML 文件的后缀是 qml ，其实就是个文本文件。下面是 一个简单的 QML 文件：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/28634543)[在CODE上查看代码片](https://code.csdn.net/snippets/378988)

1. **import** QtQuick 2.0
2. **import** QtQuick.Controls 1.1
3. **import** QtQuick.Dialogs 1.1
4. **import** an.qt.ImageProcessor 1.0
5. **import** QtQuick.Controls.Styles 1.1
7. Rectangle {
8. width: 320;
9. height: 480;
10. color: "#121212";
12. Image {
13. source: "images/IMG\_001.jpg";
14. anchors.centerIn: parent;
15. }
16. }

    这个简单的 QML 文件的开始是 import 语句，如 import QtQuick 2.0 这句，会引入 QtQuick 2.0 模块，哇，真是废话！接着废话吧。 import 和 C++ 中的 #include 类似，与 Java 中的 import 效果一样，与 JavaScript 中的……唐僧了，打住。

    Rectangle{ } 语句，定义了一个类型为 Rectangle 的对象。如果你看了《[Qt on Android: http下载与Json解析](http://blog.csdn.net/foruok/article/details/27556135" \t "_blank)》一文中有关 Json 的语法描述，应该已经知道对象要用一对花括号来描述。没错， QML 里也是这样，不过呢，花括号前要写上对象的类型。就这么简单！

    示例 QML 文档中有两个对象，一个是 Rectangle ，一个是 Image 。

    在花括号之间，是对象的属性描述（还可以有其它的，后面再说），属性是以 "property: value" 形式指定的，这点和 Json 一样。如你所见， Rectangle 对象有 width 、 color 等属性。

    属性可以分行书写，此时语句后可以不要 ";" 号，不过笔者建议 C++ 程序猿都加上 ";" ，这会避免你患上精神分裂症。当然，也可以把多个属性写在一行内，多个属性之间必须以 ";" 分割。如下所示：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/28634543)[在CODE上查看代码片](https://code.csdn.net/snippets/378988)

1. Rectangle {
2. width: 320; height: 480; color: "#121212";
3. }

    我强烈建议你不要这么干！除非有代码意外的原因，比如排版需要，比如老板觉得你代码行数太多……

**表达式**

     在《[Qt Quick 简介](http://blog.csdn.net/foruok/article/details/28620525" \t "_blank)》中笔者已经提到， QML 支持 JavaScript 表达式。比如你可以这样改写 Rectangle 对象的宽、高属性：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/28634543)[在CODE上查看代码片](https://code.csdn.net/snippets/378988)

1. Rectangle {
2. width: 23\*10;
3. height: 6\*80;
4. color: "#121212";
5. }

    我只是示意啊，你在实际项目中可别这么写，这种行为往不好听里说，有点儿脑残……当然我也可以举一个有意义的示例：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/28634543)[在CODE上查看代码片](https://code.csdn.net/snippets/378988)

1. Button {
2. text: "Quit";
3. style: ButtonStyle {
4. background: Rectangle {
5. implicitWidth: 70;
6. implicitHeight: 25;
7. border.width: control.activeFocus ? 2 : 1;
8. }
9. }
10. }

    在这个示例中我指定了按钮风格中的背景矩形，在按钮有焦点时边框宽度为 2 没有焦点时宽度为 1 。语句 "border.width: control.activeFocus ? 2 : 1" 使用了 JavaScript 的 "?:" 三元云算法（ C++ 中貌似也有……）。

    另外，慧眼如你，可能已经注意到，上面的表达式中我使用了 "control.activeFocus" ，没错，在表达式中可以引用其它对象及其属性。当你这么做的时候，待赋值的属性就和你所引用的对象的那个属性建立了关联，当被引用属性发生变化时，表达式的值会重新计算，而待赋值的属性也会变化。

    也许你心中已经有了疑问：如何引用一个对象呢？答案是：通过对象的 id 值来引用一个对象。看这里：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/28634543)[在CODE上查看代码片](https://code.csdn.net/snippets/378988)

1. Rectangle {
2. width: 320;
3. height: 480;
5. Button {
6. id: openFile;
7. text: "打开";
8. anchors.left:  parent.left;
9. anchors.leftMargin: 6;
10. anchors.top: parent.top;
11. anchors.topMargin: 6;
12. }
14. Button {
15. id: quit;
16. text: "退出";
17. anchors.left: openFile.right;
18. anchors.leftMargin: 4;
19. anchors.bottom: openFile.bottom;
20. }
21. }

    上面的示例中，退出按钮使用 id（ openFile ）引用了打开按钮。

    我的乖呀，anchors 是什么东东……先别管它，下一篇会讲到。

**注释**

    在 QML 中，注释与 C++ 中一样，单行以 "//" 开始，多行以 "/\*" 开始以 "\*/" 结束。

    注释是不被执行的，添加注释可对代码进行解释或者提高其可读性。注释同样还可用于防止代码执行，这对跟踪问题是非常有用的。

    使用注释的示例 QML ：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/28634543)[在CODE上查看代码片](https://code.csdn.net/snippets/378988)

1. /\*
2. \* the root element of QML
3. \*/
4. Rectangle {
5. width: 320;
6. height: 480;
8. Button {
9. id: quit;
10. text: "退出";
11. //use anchors to layout
12. anchors.left: openFile.right;
13. anchors.leftMargin: 4;
14. anchors.bottom: openFile.bottom;
15. //set z-order
16. z: 1;
17. }
18. }

**属性**

    其实， QML 中的属性，就是我们非常熟悉的 C++ 中的成员变量……

**属性命名**

    属性名的首字母一般以小写开始，如我们看烦了的 width 属性。

    如果属性名以多个单词表示，那么第二个及以后的单词，首字母大写。

**属性类型**

    可以在 QML 文档中使用的类型大概有三类：

* 由 QML 语言本身提供的类型
* 使用 QML 模块注册 C++ 类型
* 由 QML 模块提供的类型

    我们先看 QML 语言提供的基本类型。

**基本类型**

    QML 支持的基本类型包括整型、实数型、布尔、字符串、颜色、列表等等。这些基本类型有些是和 JavaScript 语言的基本类型对应的。

    还是之前的示例，修改了一下，通过注释标注了属性类型：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/28634543)[在CODE上查看代码片](https://code.csdn.net/snippets/378988)

1. Rectangle {
2. width: 320; //int
3. height: 480;
5. Button {
6. id: quit;
7. text: "退出"; //string
8. anchors.left: openFile.right;
9. anchors.leftMargin: 4;
10. anchors.bottom: openFile.bottom;
11. z: 1.5; // real
12. visible: **false**; //bool
13. }
14. }

    注意， QML 中属性是有类型安全检测的，也就是说你只能指定与属性类型匹配的值，否则会报错。

    请使用 Qt 助手的索引模式，以"qml basic types " 为关键字检索，找到 QML Basic Types 页面来查看完整的类型列表和每个类型的详情。

    Qt 的 QML 模块还未 QML 引入的很多 Qt 相关的类型，如 Qt 、 QtObject 、Component 、 Connections 、 Binding 等，请使用 Qt 助手检索 "qt qml qml types" 来了解。

**id 属性**

    之前在介绍表达式时提到了 id 属性，这里展开描述一下。

    一个对象的 id 属性是唯一的，在同一个 QML 文件中不同对象的 id 属性的值不能重复。当给一个对象指定了 id ，就可以在其它对象或脚本中通过 id 来引用该对象。在“表达式”一节中我们已经演示了如何通过 id 来引用一个对象。

    请注意， id 属性的值，首字符必须是小写字母或下划线并且不能包含字母、数字、下划线以外的字符。

**列表属性**

    列表属性类似于下面这样：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/28634543)[在CODE上查看代码片](https://code.csdn.net/snippets/378988)

1. Item {
2. children:[
3. Image{},
4. Text{}
5. ]
6. }

    列表是包含在方括号内，以逗号分隔的列表元素。看起来是不是挺熟悉？在《[Qt on Android: http下载与Json解析](http://blog.csdn.net/foruok/article/details/27556135" \t "_blank)》中，我们举过 Json 数组的例子，再看看：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/28634543)[在CODE上查看代码片](https://code.csdn.net/snippets/378988)

1. [
2. "name":"zhangsan",
3. {
4. "age":30,
5. "phone":"13588888888",
6. "other": ["xian", **null**, 1.0, 28]
7. }
8. ]

    其实列表和 JavaScript 的数组是类似的，其访问方式也一样：

* length 属性提供了列表内元素的个数
* 列表内的元素通过数组下标来访问（[index]）

    值得注意的是，列表内只能包含 QML 对象，不能包含任何基本类型（如整型、布尔型）。这点与 Json 是不一样的。下面是访问列表的示例：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/28634543)[在CODE上查看代码片](https://code.csdn.net/snippets/378988)

1. Item {
2. children:[
3. Text{
4. text: "textOne";
5. },
6. Text{
7. text: "textTwo";
8. }
9. ]
10. Component.onCompleted:{
11. **for** (**var** i = 0; i < children.length; i++)
12. console.log("text of label ", i, " : ", children[i].text)
13. }
14. }

   如果你一个列表内只有一个元素，也可以省略方括号。如下所示：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/28634543)[在CODE上查看代码片](https://code.csdn.net/snippets/378988)

1. Item {
2. children:Image{}
3. }

    不过笔者还是建议你始终使用方括号，哪怕其中只有一个元素。

    有没有什么问题？有就要说啊，闷在心里会憋坏自己的。好吧，你不说我就说了。在我们访问列表的示例中出现了一个新的内容，Component.onCompleted :{} ，这是什么东东呢？接下来我们就来讲它。

**信号处理器**

    信号处理器，其实等价于 Qt 中的槽。但是我们没有看到类似 C++ 中的明确定义的函数……没错，就是这样，你的的确确只看到了一对花括号！对啦，这是 JavaScript 中的代码块。其实呢，你可以理解为它是一个匿名函数。而 JavaScript 中的函数，其实具名的代码块。函数的好处是你可以在其它地方根据名字调用它，而代码块的好处是，除了定义它的地方，没人能调用它，一句话，它是私有的。代码块就是一系列语句的组合，它的作用就是使语句序列一起执行。

    让我们回头再看信号处理器，它的名字还有点儿特别，一般是 on{Signal} 这种形式。比如 Qt Quick 中的 Button 元素有一个信号 clicked() ，那么你要可能会写出这样的代码：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/28634543)[在CODE上查看代码片](https://code.csdn.net/snippets/378988)

1. Rectangle {
2. width: 320;
3. height: 480;
5. Button {
6. id: quit;
7. text: "退出";
8. anchors.left: parent.left;
9. anchors.leftMargin: 4;
10. anchors.bottom: parent.bottom;
11. anchors.bottomMargin: 4;
12. onClicked: {
13. Qt.quit();
14. }
15. }
16. }

    上面的 QML 代码其实已经是一个简单 QML 应用了，这个应用在窗口的左下角放了个退出按钮，当用户点击它时会触发按钮的 clicked() 信号，而我们定义了信号处理器来响应 clicked() 信号——调用 Qt.quit() 退出应用。

    你看到了，当信号是 clicked() 时，信号处理器就命名为 onClicked 。就这么简单，以 on 起始后跟信号名字（第一个字母大写）。

**分组属性**

    在某些情况下使用一个 '.' 符号或分组符号把相关的属性形成一个逻辑组。分组属性可写以下这样：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/28634543)[在CODE上查看代码片](https://code.csdn.net/snippets/378988)

1. Text {
2. font.pixelSize: 18;
3. font.bold: **true**;
4. }

    也可以这样写：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/28634543)[在CODE上查看代码片](https://code.csdn.net/snippets/378988)

1. Text {
2. font { pixelSize: 12; bold: **true**; }
3. }

    其实呢，可以这么理解，font 属性的类型本身是一个对象，这个对象又有 pixelSize / bold / italic / underline 等等属性。对于类型为对象的属性值，可以使用 "." 操作符展开对象的每一个成员对其赋值，也可以通过分组符号（一对花括号）把要赋值的成员放在一起给它们赋值。对于后者，其形式就和对象的定义一样了，起码看起来木有区别。所以呢，又可以这么理解上面的示例： Text 对象内聚合了 font 对象。 OK ，就是聚合。

**附加属性**

    属性真难搞！到现在还没讲完，不但你烦了，我也快坐不住了。我保证，这是最后一个要点了，不过也是最复杂最难以理解的属性了。对于这种玩意儿，我一向的做法时，不能理解的话就接受，你就当它生来如此，存在即合理，只要学会怎么用它就 OK 了。

    在 QML 语言的语法中，有一个附加属性（attached properties）和附加信号处理器（attached signal handlers）的概念，这是附加到一个对象上的额外的属性。从本质上讲，这些属性是由附加类型（attaching type）来实现和提供的，它们可能被附加到另一种类型的对象上。附加属性与普通属性的区别在于，对象的普通属性是由对象本身或其基类（或沿继承层级向上追溯的祖先们）提供的。

   举个例子，下面的 Item 对象使用了附加属性和附加信号处理器：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/28634543)[在CODE上查看代码片](https://code.csdn.net/snippets/378988)

1. **import** QtQuick 2.0
3. Item {
4. width: 100;
5. height: 100;
7. focus: **true**;
8. Keys.enabled: **false**;
9. Keys.onReturnPressed: console.log("Return key was pressed");
10. }

    你看， Item 对象可以访问和设置 Keys.enabled 和 Keys.onReturnPressed 的值。 enabled 是 Keys 对象的一个属性。 onReturnPressed 其实是 Keys 对象的一个信号。对于附加信号处理器，和前面讲到的普通信号处理器又有所不同。普通信号处理器，你先要知道信号名字，然后按照 on{Signal} 的语法来定义信号处理器的名字；而附加信号处理器，你只要通过附加类型名字引用它，把代码块赋值给它即可。

    最后说下 Keys 对象，它是 Qt Quick 提供的，专门供 Item 处理按键事件的对象。它定义了很多针对特定按键的信号，比如上面的 onReturnPressed ，还定义了更为普通的 onPressed 和 onReleased 信号，一般地，你可以使用这两个信号来处理按键（请对照 Qt C++ 中的 keyPressEvent 和 keyReleaseEvent 来理解）。它们有一个名字是 event 的 KeyEvent 参数，包含了按键的详细信息。如果一个按键被处理， event.accepted 应该被设置为 true 以免它被继续传递。

   下面是使用 onPressed 信号的一个示例，它检测了左方向键：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/28634543)[在CODE上查看代码片](https://code.csdn.net/snippets/378988)

1. Item {
2. anchors.fill: parent;
3. focus: **true**;
4. Keys.onPressed: {
5. **if** (event.key == Qt.Key\_Left) {
6. console.log("move left");
7. event.accepted = **true**;
8. }
9. }
10. }

    版权所有 foruok，转载请注明出处：<http://blog.csdn.net/foruok>

    好啦，关于 QML 语言的基础性介绍就到这里，相信现在你已经可以看懂简单的 QML 文档了。有的同学可能有疑问了，这节还有一些东东只见用不见讲啊，比如 Rectangle / Text / Image / Item / Button / Component / Qt 等等，抱歉，现在只能揣着糊涂装明白了，下一篇我们会讲这些东西。