EXML基本语法(一)

EXML是一种严格遵循XML语法的标记语言，通常用于描述静态UI界面。这节内容将详细介绍EXML的语法规则。

根节点

首先分析一个最简单EXML文件内容：

```
<e:Group class="app.MyGroup" xmlns:e="http://ns.egret.com/eui">
</e:Group>
```

它跟XML一样，是由标签组成的。每个标签都有个命名空间前缀，例如 `<e:Group>` 中的 `e` ,它的对应命名空间声明也在根节点上： `xmlns:e="http://ns.egret.com/eui"` 。以e这个命名空间开头的节点，表示在EUI这个UI库中的组件。而 `<e:Group>` 中的 `Group` 就是对应代码中 `eui.Group` 这个类。

这个例子中只有一个根节点，根节点上的 `class` 属性表示它在运行时解析后要注册为的全局类名。以上的EXML文件，在运行时解析后完全等价于如下代码：

```
module app {
    export class MyGroup extends eui.Group {
        public constructor(){
            super();
        }
    }
}
```

从这个例子可以看出EXML文件与代码的对应关系。EXML解析后会变成一个自定义类，继承的父类就是EXML的根节点，模块名和类名定义在跟节点上的 `class` 属性内。

`注意:一定要加class这个属性，否则编译会报错`

添加子项

上面的例子只有一个根节点，将它扩展，添加一个Image子项：

```
<e:Group class="app.MyGroup" xmlns:e="http://ns.egret.com/eui">    <e:Image />
</e:Group>
```

以上内容等价于如下代码：

```
module app {
    export class MyGroup extends eui.Group {
        public constructor(){
            super();
            var image = new eui.Image();
            this.addChild(image);
        }
    }
}
```

设置属性

刚刚的例子只添加了一个空图片，什么都显示不出来，接下来给它设置一些属性：

```
<e:Group class="app.MyGroup" xmlns:e="http://ns.egret.com/eui">
    <e:Image source="image/button_up.png" x="10"/>
</e:Group>
```

上述代码直接给Image节点加了source属性，它解析后相当于如下代码：

```
var image = eui.Image();
image.source = "image/button_up.png";
image.x = 10;
this.addChild(image);
```

声明属性时不用考虑值的类型，都写在双引号内即可。解析器运行时会去读取节点上这个属性的类型，并正确格式化为对应的结果。例如source的结果是复制一个字符串，而x的结果是赋值一个数字10，不会带双引号。

以上是最常见的属性写法，通常只能描述简单数据类型的赋值，如果是复杂数据类型，比如要对属性赋值为另一个节点时，可以采用另一种写法：

```
<e:Group class="app.MyGroup" xmlns:e="http://ns.egret.com/eui">
    <e:Image source="image/button_up.png" x="10"> <e:scale9Grid> <e:Rectangle x="10"
 y="10" width="45" height="35"/> </e:scale9Grid>
    </e:Image>
</e:Group>
```

`<e:scale9Grid>` 是属性节点，表示父级节点Image的scale9Grid属性，这个属性要接受一个Rectangle对象的实例。以上内容等价为：

```
var image = eui.Image();
image.source = "image/button_up.png";
image.x = 10;
var rect = new Rectangle();
rect.x = 10;
rect.y = 10;
rect.width = 45;
rect.height = 35;
image.scale9Grid = rect;
this.addChild(image);
```

ID属性

我们可以在节点上声明一个id属性，注意这个id属性与HTML中的id并不是一回事，它的结果相当于给解析后的类声明了一个公开变量。例如：

```
<e:Group class="app.MyGroup" xmlns:e="http://ns.egret.com/eui">
    <e:Image id="iconDisplay" />
</e:Group>
```

等价于：

```
module app {
    export class MyGroup extends eui.Group {
        public iconDisplay:eui.Image;
        public constructor(){
            super();
            var image = new eui.Image();
            this.addChild(image);
            this.iconDisplay = image;
        }
    }
}
```

属性语法糖

前面的例子描述了如何给Image设置scale9Grid(九宫格)属性，使用起来有些麻烦。因此准备了一系列的语法糖可以简化属性的声明，九宫格的属性还可以用如下方式声明：

```
<e:Group class="app.MyGroup" xmlns:e="http://ns.egret.com/eui">
    <e:Image source="image/button_up.png" x="10" scale9Grid="10,10,45,35" />
</e:Group>
```

对于特别常用的属性，解析器会内置一些语法糖，使编写更加简洁。这里介绍另一个语法糖，width或height属性可以直接写百分比：

```
<e:Group class="app.MyGroup" xmlns:e="http://ns.egret.com/eui">
    <e:Image source="image/button_up.png" width=“100%” height="100%" />
</e:Group>
```

解析器在处理这两个属性时，若赋值的是具体数字，按默认规则走，生成对width/height的赋值代码，若赋值的是百分比字符串，那么就会生成percentWidth等属性的赋值代码：

```
var image = eui.Image();
image.source = "image/button_up.png";
image.percentWidth = 100;
image.percentHeight = 100;
this.addChild(image);
```

所以代码中，实际上应该使用percentWidth等属性去设置百分比，width属性在代码中只接受具体的数字。
目前只有以上这两种属性语法糖需要注意，不排除以后把使用频率高的属性也设计为语法糖。

节点默认属性

上文介绍了复杂属性节点的声明方式，要先显式声明一个属性名称的节点，内部再跟上要赋值的节点。这里还有一个类似语法糖的写法，eui库内的组件，通常都会有一个默认属性，如果子节点是赋值给父节点的默认属性，那么可以省略属性名节点。如下例：

```
<e:Scroller class="app.MyScroller" xmlns:e="http://ns.egret.com/eui">
    <e:viewport>
        <e:Group/>
    </e:viewport>
</e:Scroller>
```

这个例子中，将Group实例赋值给了一个滚动容器Scroller的viewport属性。由于viewport是Scroller的默认属性，因此我们可以直接省略 `<e:viewport>` 节点，改成如下写法：

```
<e:Scroller class="app.MyScroller" xmlns:e="http://ns.egret.com/eui">
    <e:Group/>
</e:Scroller>
```

除了支持省略属性名节点外，若默认属性的类型是一个数组，还可以省略Array节点。其实添加子项也只是省略默认属性的一种特例，因为容器的默认属性是 `elementsContent` ,类型正是数组。最开始添加子项的那个例子完整写法如下：

```
<e:Group class="app.MyGroup" xmlns:e="http://ns.egret.com/eui">
    <e:elementsContent>
        <e:Array>
            <e:Image/>
        </e:Array>
    </e:elementsContent>
</e:Group>
```

当然，直接用最简洁的省略默认属性写法即可。