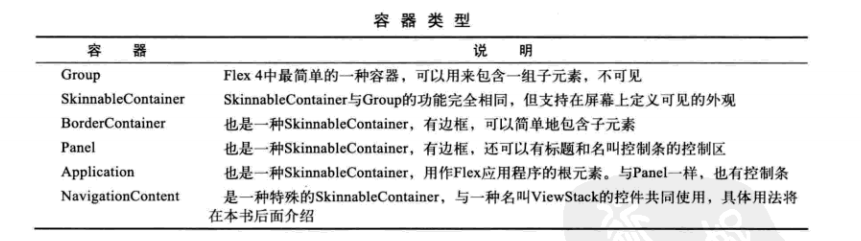
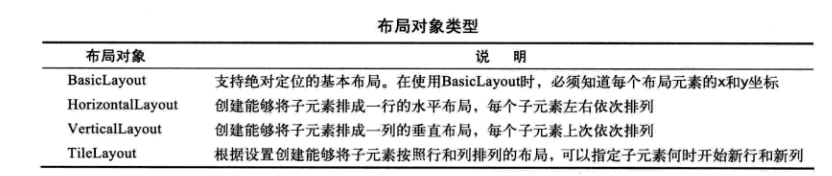
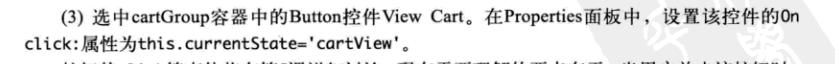
# 容器

布局对象

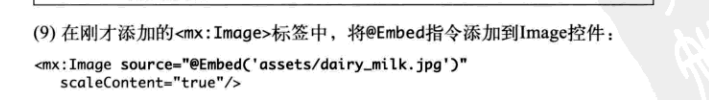


# 状态的切换

this.currentState='cartView'



# Image使用编译的资源



# Model数据模型和绑定

1：写在页面中

<fx:Declarations>

<!-- 将非可视元素（例如服务、值对象）放在此处 -->

<fx:Model id="groceryInventory1">

<groceries>

<catName>Dairy</catName>

<prodName>Milk</prodName>

<imageName>assets/dairy\_milk.jpg</imageName>

<cost>1.20</cost>

<listPrice>1.99</listPrice>

<isOrganic>true</isOrganic>

<isLowFat>true</isLowFat>

<description>Direct from California where cows are happiest!</description>

</groceries>

</fx:Model>

</fx:Declarations>

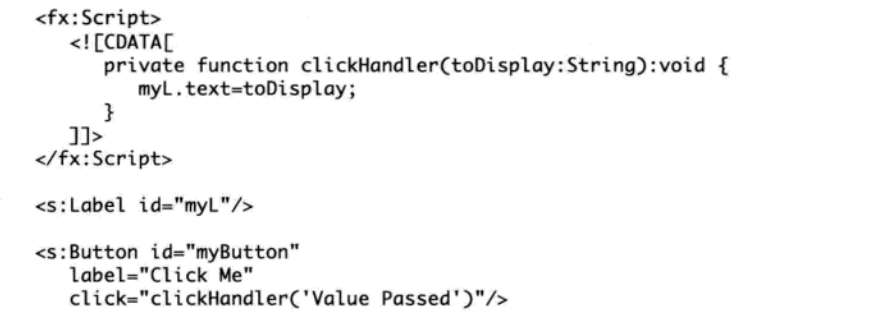
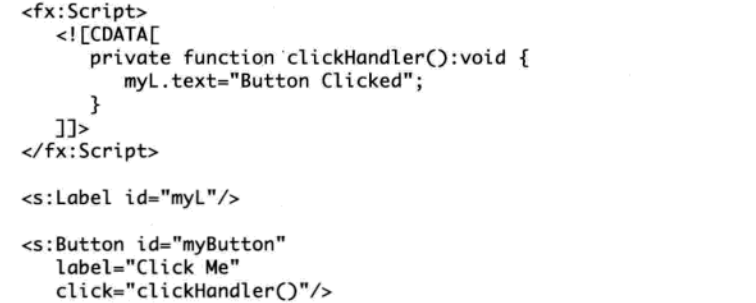
2：写在外部文件

<fx:Model id="groceryInventory" source="assets/inventory.xml"></fx:Model>

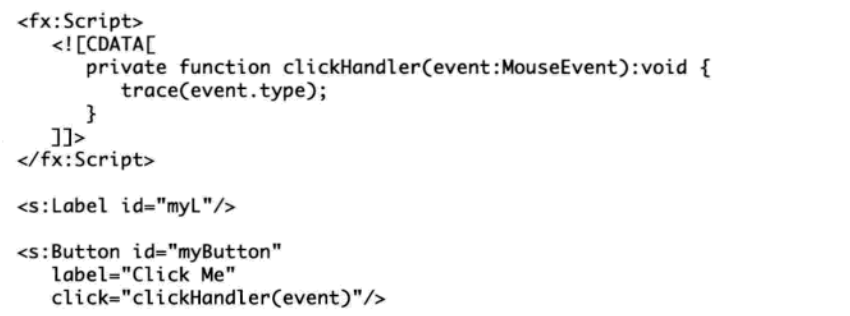
**使用时候，在具体空间中 text使用 {groceryInventory.description}**

# flex的函数体

1：不带参数 2 带参数



2：鼠标事件



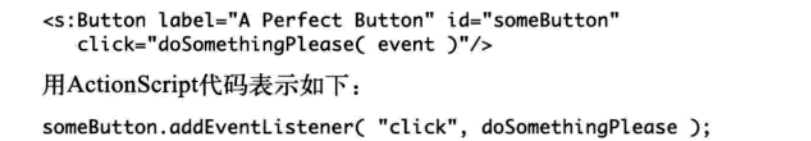
# flex事件

## 系统事件

creationComplete:组件完成初始化

## 用户事件

## 监听事件

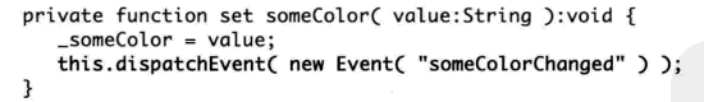


## 分发事件

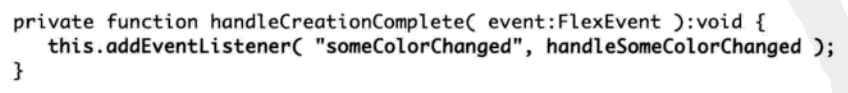


自定义的事件监听和分发示例：

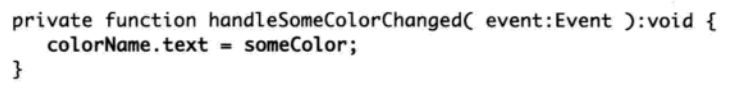
（设置事件分发）

\shesd

（设置事件监听）

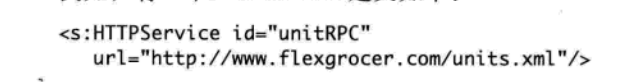


(执行监听事件)



# Flex调用远程xml数据

方式1:lastResult





方式2：resul

creationComplete="handleCreationComplete(event)" currentState="State1">

<s:HTTPService id="categoryService" url="http://www.flexgrocer.com/category.xml"

result="handleCategoryResult(event)"/>

private function handleCreationComplete(event:FlexEvent):void

{

categoryService.send();//载入url中的xml

}

private function handleCategoryResult(event:ResultEvent):void

{

//dosomething

}

# 数据

## 基本类

Boolean 布尔

Int 整数

Number 数值

Object 对象

String 字符串

Uint自然数

Array 数组

Date 日期

Error错误性息

Function 函数

RegExp 正则表达式

Xml XML对象

XmlList XML数组

## E4X

category.product.(@cost == "123")

category.product.(@cost == "123").(unit ==”each”)

不论何处的product都返回

category..product

示例：

<s:HTTPService id="categoryService"

url="http://www.flexgrocer.com/category.xml"

resultFormat="e4x"

result="handleCategoryResult(event)"/>

[Bindable]

private var categories:XMLListCollection;

private function handleViewCartClick( event:MouseEvent ):void {

this.currentState="cartView";

}

private function handleCreationComplete(event:FlexEvent):void {

categoryService.send();

}

private function handleCategoryResult(event:ResultEvent):void{

categories = new XMLListCollection(event.result.category);

}

# 数据绑定

<s:List includeIn="State1" x="200" y="40" dataProvider="{categories}" labelField="name">

<s:layout>

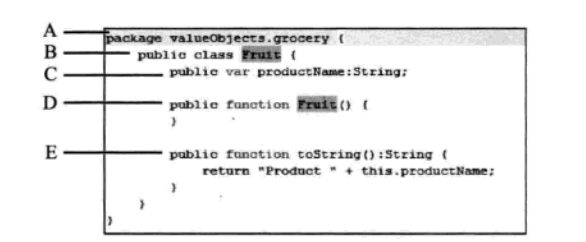
<s:HorizontalLayout/>

</s:layout>

</s:List>

# ActionScript 类

## 示例



A :保存的路径，类似 命名空间

B: 类名和文件名

C: 类的属性

D：构造函数

E：方法

## 类修饰符

Public

Internal

Dynamic

Final => 类似抽象类

## 成员修饰符

Internal（默认）：同一包（命名空间）有效

Private 私有的：同一类有效

Protected 保护的 同一类和派生类有效

Public

Static

Override 覆盖基类的方法

Final 禁止基类覆盖方法

## 属性（注意as类的属性有些特别，举例）

Public class Car

{

Private var \_name:string =”undefined”;

Public function get name():string

{

Return \_name;

}

Public function set name(newName): void

{

\_name = newName;

}

}

## 继承

继承使用关键字 extents

## AS 核心类

（可以参考帮助文档）

### Char

### Sting

### Array

### XML

XML的赋值（不使用引号和结束的分号）

Var res ：xml=

<root>

<item>

<title>学习flex<title><content>flex是一个richAPI<content>

</item>

<item>

<title>学习flex<title><content>flex是一个richAPI<content>

</item>

</root>

## AS 内置类

### Timer类

public function testTimer():void

{

//创建一个1000毫秒一次，执行5次的定时器

var NewTimer :Timer = new Timer(1000,5);

//为定时器间隔和完成添加监听器

NewTimer.addEventListener(TimerEvent.TIMER,OnTick);

NewTimer.addEventListener(TimerEvent.TIMER\_COMPLETE,OnTimeComplete);

//启动定时器

NewTimer.start();

}

private function OnTick():void

{

//dosomething

}

private function OnTimeComplete():void

{

//dosomething

}

### Math类

### Socket类

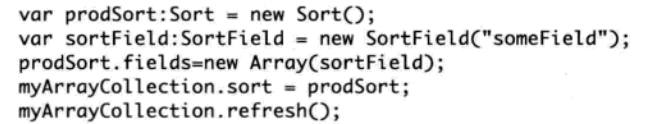
### Alert类

# ActionScript 基本语法和C#的区别

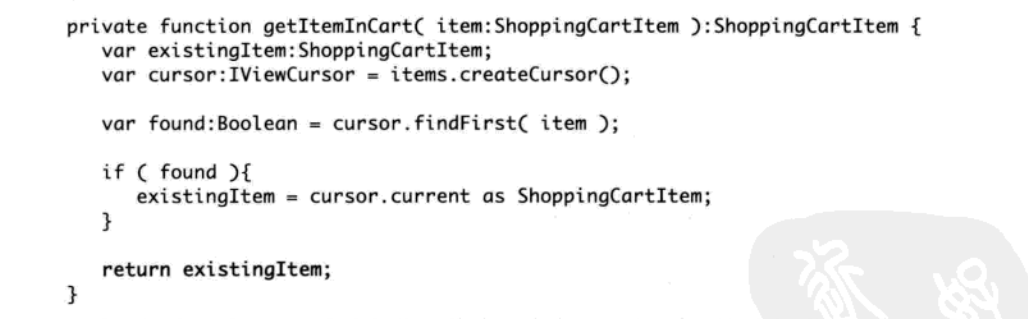
## Fore each



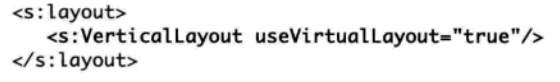
## 排序



## 光标



## 虚拟化



## 拖拽

