ActionScript 3.0 CookBook

中文翻译

可公布

作者：Joey Lott, DarronSchall, Keith Peters
译者：常青(李新业)
电子邮件：Xinye0123@gmail.com
博客：http://blog.csdn.net/lixinye0123
论坛：http://groups.google.com/group/AS3_CN

注：因本人水平有限，出现理解错误或翻译不妥之处在所难免，恳请读者批评指正，在阅读中如遇到不解的问题，可到http://groups.google.com/group/AS3_CN留言。

推荐官方阅读器： Adobe Reader 8

感谢 www.linuxfans.org 提供下载支持！

1.0. ActionScript 3.0 Cookbook 概述

概述

在Ajax 和 微软 WPF 袭来之前，Macromedia 率先推出基于Flash的RIA解决方案，用于创建具有桌面程序富有交互和多功能的Web应用程序，我们称之为“Rich Internet Application”。现在，新东家 Adobe 更是赋予了Flash超越Web之能力，使之成为完整的开发环境。
除了理论，本书来源于实际ActionScript应用，超过300个解决方法用于解决各种编写代码中遇到的问题。比如：

检测用户Flash 播放器版本或操作系统

开发自定义类

数据和类型格式化

字符串的使用

开发用户界面组件

声音和视频的使用

使用Flash Remoting and web services 实现远程过程调用

读取，发送和搜索XML数据

更多。。。。。。

1.1. 新建一个 ActionScript 工程

问题

运行 Flex Builder 2 然后新建一个ActionScript 工程

解决方法

使用 新建ActionScript 工程向导来建立你的工程

讨论

一个ActionScript 工程一般至少包含一个class文件和一个用来放置编译输出的SWF和HTML的bin目录。还包含一些设定来告诉编译器怎么去编译工程。我们只要用向导创建项目，其他的事都交给Flex Builder 2 维护就行了。有几种方法启动向导，你可以使用菜单中的File ->New->ActionScript Project, 或者点击左上方的New 按钮，然后从列表中选择 ActionScript Project 。

打开先导后选择 New ActionScript Project ，下一步输入工程名称，比如 *ExampleApplication*, 当你创建好工程后，我们会看到主应用程序文件被设置成工程名加上 *.as 扩展名*。

点击下一步，这里可以添加自定义类，额外的库，或者指定输出目录名称来代替默认的bin目录，不过现在我们不必关心这些，点击Finish完成向导。

接着Flex Builder 2 为我们做好了一切。在Navigator 视图上我们看到 *ExampleApplication* 工程, 包含了一个空的bin目录和一个*ExampleApplication.as* 类文件。注意创建的主类文件已经在编辑视图中打开了，而且在Outline 视图我们看到一个类树型结构，它包含了类方法，属性和import 语句。

运行我们的程序，只要点击上面的绿色的三角形图标按钮，右边的小虫图标的按钮是调试按钮, 两个命令都是生成.swf 和 html 文件，它会启动浏览器自动运行我们的程序。

到现在我们还没有添加任何语句，这相当于在Flash IDE中一个空白的 *.fla* 文件一样, 当运行的时候除了蓝色的背景什么也没有。

1.2. 自定义应用程序属性

问题

我要改变SWF的尺寸或背景颜色

解决方法

指定项目属性里的编译器参数或者class文件的 metadata 。

讨论

不像早期版本的Flash，ActionScript 3.0 编译器真正是一个命令行编译器。你可以通过命令行再加上一长串参数来创建类或目录，在eclipse里让这一切变得更简单些。

当建立ActionScript 工程后，默认情况下会生成500x375 尺寸的 .swf, 帧速为24/秒，背景色为蓝色。我们可以改变这些设定，有几种方法。

第一种方法就是通过ActionScript编译器参数改变编译器设定。右键点击工程，在菜单中选择 Properties ，然后在左边选择ActionScript Compiler，在右边找到"Additional compiler arguments." 在这里就可以输入参数了，下面是些常见的参数

-default-size width height

-default-background-color color

-default-frame-rate fps

可以这样写：

-default-size 800 600

-default-background-color 0xffffff

-default-frame-rate 31

第一个参数设定输出swf尺寸为800x600 象素. 第二个参数设定背景色为白色，第三个参数设定播放帧速为31帧每秒。多个参数可以这样写：

-default-size 800 600 -default-frame-rate 31

第二种方法就是通过类文件种的metadata 来改变设定。Metadata 中包含的语句不会立即被解释，但是在编译的时候编译会去检测。下面的语句具有等同效果

```
[SWF(width="800", height="600", backgroundColor="#ffffff", frameRate="31")]
```

这一行语句放在import之后，类定义之前，如：

```
package ...{
    import flash.display.Sprite;
    [SWF(width="800", height="600", backgroundColor="#ffffff", frameRate="31")]
    public class ExampleApplication extends Sprite
    {
        public function ExampleApplication()
        {
        }
    }
}
```

1.3. 在哪里写ActionScript 代码呢

问题

当你有了ActionScript工程后，接着就需要知道任何输入代码。

解决方法

在类结构中或方法体中添加 ActionScript 代码

讨论

在 以前的ActionScript 1.0 和 2.0中, 有多种途径添加代码：在时间线上，按钮上或电影剪辑上, 在电影剪辑的时间线上通过＃include命令引入外部的as文件或class文件。但是 ActionScript 3.0 是完全基于类的，所以所有的代码都必须放置在类文件中。

当你创建一个新的 ActionScript 工程后，主类文件被自动创建，并且在代码视图中代开了，刚开始的代码大概是这样的：

```
package ...{
    import flash.display.Sprite;

    public class ExampleApplication extends Sprite
    ...{
        public function ExampleApplication( )
        ...{

        }
    }
}
```


可能你很熟悉 ActionScript 2.0中的类, 但是3.0发生了很多变化，这些我们将在第二章讨论，在这里先学完基础概念先。

首先注意到代码顶层有个关键字 ***package*** ，Packages（包） 是用来组织一群相关联的类文件的 。在 ActionScript 2.0, 包是用来判断类文件的路径。在 ActionScript 3.0 中必须指定包，例如，我们有个utility类包，要这样申明：

```
package com.as3cb.utils {}
```

如果你不指明包名，那么该类就输入最顶层的默认包。

接下来，加入 **import** 语句，引入一个类就相当于在当前的代码文件中创建了使用该类的快捷方式，这样我们就不需要输入全路径来使用它了。例如，你可以使用下面的 import 语句：

```
import com.as3cb.utils.StringUtils;
```

这样我们就可以直接引用 *StringUtils* 这个类了。从*flash.display* 引入*Sprite* 类是因为默认的类文件继承了*Sprite* 类。

接下来就看到我们的主类 ExampleApplication，注意到在class关键字前有个关键字*public* ，表明该类是共有的。最后有个公共方法，方法名和主类一样，这种方法称为构造器，当一个类实例被创建时，其构造器会被自动执行，在这里，当swf文件被 Flash 播放器载入时构造器就会被执行。

```
package ...{
    import flash.display.Sprite;
    public class ExampleApplication extends Sprite ...{
        public function ExampleApplication( ) ...{
            graphics.lineStyle(1, 0, 1);
            for(var i:int=0;i<100;i++) ...{
                graphics.lineTo(Math.random( ) * 400, Math.random( ) * 400);
            }
        }
    }
}
```

保存然后运行程序，浏览器会打开一个html文件，显示一个swf里画了100条随即直线。正如你所看到的，当swf被播放器载入后构造器就会被执行。

在这里联系中，我们把代码直接写在了构造器中，但时最好的办法是在构造器中引用一个方法，在这个方法中添加代码.

对于新手来说，现在你已经学会了如何添加代码了。

1.4. 如何跟踪信息

问题

你需要在运行时跟踪信息或某个数据变量

解决办法

使用*trace*函数，把数据传给它，运行程序，你会发现信息已经在Eclipse的控制台下输出了。

讨论

你可以跟踪一个消息或一个变量的值，也可以输出任何其他数据，就像你在早期的版本中那样，比如：

```
trace("Hello, world");

trace(userName);
```

```
trace("My name is " + userName + ".");
```

一旦swf在外部浏览器里运行，就没办法捕获trace输出的信息了，幸运的是Flex Builder2有Console视图，Console视图就相当于Flash的Ouput面板。

需要注意的是使用trace则必须在调试模式下运行程序，这样才能在Console视图显示数据，下面的代码创建一个变量，然后赋值，然后用trace输出。

```
package {
    import flash.display.Sprite;
    public class ExampleApplication extends Sprite {
        public function ExampleApplication(    ) {
            var userName:String = "Bill Smith";
            trace("My name is " + userName + ".");
        }
    }
}
```

现在在调试模式下运行程序，运行完关闭浏览器，你会看到在Eclipse下已经输出数据了。

要运行debug版本的程序，必须要安装debug版本的Flash播放器。否则会显示错误信息，另外debug版本的播放器可以把输出信息到一个文件上，查找mm.cfg文件。一般信息如下：

Operating system	Location
Windows XP	C:\Documents and Settings\[user name]\mm.cfg
Windows 2000	C:\mm.cfg
Mac OS X	MacHD:Library:Application Support:macromedia:mm.cfg

mm.cfg 文件允许你设置如下变量：
TraceOutputFileEnable

设置值为 0 (不写入文件) 或 1 (写入文件).

TraceOutputFileName

文件路径，如果没有指定，会在mm.cfg的同目录下生成一个叫flashlog.txt文件

ErrorReportingEnable

设置值为 0 (不输出错误信息到文件) 或 1 (输出错误信息). 默认为0

MaxWarnings

写入文件的错误信息数量。如果为0则没有限制。

例子：

TraceOutputFileEnable=1

TraceOutputFileName=C:\flex.log

1.5. 处理事件

问题

我要重复执行某段代码

解决办法

在*enterFrame*事件中添加监听器和关联处理方法

讨论

在 ActionScript 2.0 中处理 *enterFrame* 事件是很简单的，你只要创建时间线函数调用 *onEnterFrame* 然后每次新帧开始时就会自动调用。在 ActionScript 3.0 中有各种各样的事件需要控制，访问他们也是不难的。

如果你熟悉ActionScript 2.0中的*EventDispatcher* 类的话，你就很好理解 ActionScript 3.0 事件句柄了。要广播 *enterFrame* 事件，你要告诉你的程序去监听这个事件然后指定回调函数。用 *addEventListener* 方法可以做到：

```
addEventListener(type:String, listener:Function)
```

type 参数指出你要监听的事件类型，比如"enterFrame". 然而自己输入这些字符串容易出错，最好的办法就是调用 *Event* 类的静态成员属性：导入*Event* 类，调用*addEventListener* 方法：

```
addEventListener(Event.ENTER_FRAME, onEnterFrame);
```

第二个参数onEnterFrame,指向类中定义的回调函数，该函数需要传递进EVENT的一个实例：

```
import flash.events.Event;
private function onEnterFrame(event:Event) { }
```

event 对象包含一些于该事件有关的信息。这里有个简单的例子：画出一些随机线。

```
package {
    import flash.display.Sprite;
    import flash.events.Event;
    public class ExampleApplication extends Sprite {
        public function ExampleApplication(    ) {
            graphics.lineStyle(1, 0, 1);
```

```
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
        }
        private function onEnterFrame(event:Event):void {
            graphics.lineTo(Math.random(    ) * 400, Math.random(    ) * 400);
        }   }   }
```

1.6. 响应鼠标和键盘事件

问题

我要处理鼠标或键盘事件

解决办法

监听和捕获处理鼠标和键盘事件

讨论

处理鼠标和键盘事件很类似于*enterFrame* 事件，这些在1.5节已经讨论过，只是略有不同。对于鼠标事件，主程序不会直接接收，需要通过一个可视组件监听它（关于可视组件会在第5章讨论）。下面的例子创建一个sprite，添加到可视组件列，然后在它上面画了个矩形：

```
package {
    import flash.display.Sprite;
    import flash.events.MouseEvent;
    public class ExampleApplication extends Sprite {
        private var _sprite:Sprite;
        public function ExampleApplication(    ) {
            _sprite = new Sprite(    );
            addChild(_sprite);
            _sprite.graphics.beginFill(0xffffff);
            _sprite.graphics.drawRect(0, 0, 400, 400);
            _sprite.graphics.endFill(    );
```

注意：鼠标事件名称被定义在 *MouseEvent* 类中，事件处理函数需要传递进一个*MouseEvent* 类实例，现在为sprite加入鼠标监听器：

```
            _sprite.addEventListener(MouseEvent.MOUSE_DOWN, onMouseDown);
            _sprite.addEventListener(MouseEvent.MOUSE_UP, onMouseUp);
        }
```

接着，定义两个处理函数onMouseDown 和 onMouseUp:

```
        private function onMouseDown(event:MouseEvent):void {
            _sprite.graphics.lineStyle(1, 0, 1);
            _sprite.graphics.moveTo(mouseX, mouseY);
            _sprite.addEventListener(MouseEvent.MOUSE_MOVE, onMouseMove);
        }
        private function onMouseUp(event:MouseEvent):void {
            _sprite.removeEventListener(MouseEvent.MOUSE_MOVE, onMouseMove);
        }
```

onMouseDown 方法设置画线的类型，移动画刷到鼠标点击位置，然后添加了第三个鼠标监听器监听 *MouseMove* 事件

onMouseUp 方法用*removeEventListener* 方法移除监听器，它和*addEventListener* 方法具有相同语法结构，只是作用相反罢了

最后，定义onMouseMove 函数

```
        private function onMouseMove(event:MouseEvent):void {
            _sprite.graphics.lineTo(mouseX, mouseY);
        }
    }
}
```

这样就建立了一个事件驱动的绘画程序。

键盘事件的处理简单一些，只需要监听和响应键盘事件，接受这些事件的对象必须出于激活状态。我们需要在主程序中加入这一行：

```
stage.focus = this;
```

下面的例子展示一个简单的类，它监听键盘的 *keyDown* 事件，输出按键的字符码，

```
package {
    import flash.display.Sprite;
    import flash.events.KeyboardEvent;
    public class ExampleApplication extends Sprite {
        public function ExampleApplication(    ) {
            stage.focus = this;
            addEventListener(KeyboardEvent.KEY_DOWN, onKeyDown);
        }
```

```
            private function onKeyDown(event:KeyboardEvent):void {
                trace("key down: " + event.charCode);
            }
        }
}
```

1.7. 算术运算

问题

我要修改一些东西，比如sprite的角度和位置

解决办法

使用复合赋值运算来改变变量或属性的值

讨论

经常我们需要的新值需要建立在旧值的基础上，比如说，我要移动sprite到离当前位置向右10个像素的地方。

一条赋值语句通过赋值操作符（=号）建立，=号右边表达式被运算出来然后其结果存储在左边的变量或属性中。

```
// 给变量 quantity 增加6
quantity = quantity + 6;
```

在这些算术操作中，还有些很便利的复合操作符，如 +=, -=, *=, and /= 会被经常用到。

下面两个语句都是给quantity 加上6:

```
quantity = quantity + 6;
quantity += 6;
```

下面两个语句都是给quantity 减去6:

```
quantity = quantity - 6;
quantity -= 6;
```

下面两个语句让quantity 乘以 factor:

```
quantity = quantity * factor;
quantity *= factor;
```

下面两个语句让 quantity 除以 factor:

```
quantity = quantity / factor;
quantity /= factor;
```

如果只是增加1或减少1，还可以象下面这样写

这个语句让quantity 增加1:

```
quantity++;
```

下面的两个语句效果相同

```
quantity += 1;
```

这个语句让 quantity 减去1:

```
quantity --;
```

下面的两个语句效果相同：

```
quantity = quantity   1;
quantity -= 1;
```

自增和自减运算符还有前缀写法：

```
var quantity:Number = 5;
trace(quantity++);   // Displays: 5
trace(quantity);      // Displays: 6
var quantity:Number = 5;
trace(++quantity);   // Displays: 6
trace(quantity);      // Displays: 6
```

回到起初的问题，你可以用这些操作符修改属性值。下面的代码指定了sprite每帧角度加5：

```
private function onEnterFrame(event:Event) {
   _sprite.rotation += 5;
}
```

1.8. 逻辑运算

问题

我想检测两个值的大小

解决办法

使用==号来比较两个值，使用*isNaN()*来检测是否是有效值.

讨论

==号表达式总是返回布尔值来表示两个值是否相等。当两个数类型不同时，比较时会自动转换为相同的类型再进行比较，如字符型的6和数字型的6比较的话被认为相等。

```
trace(5 == 6);       // : false
trace(6 == 6);       // : true
trace(6 == "6");     // : true
trace(5 == "6");     // : false
```

默认的工程项目，在运行上面的代码会出错。因为编译器被设置为强类型编译检测。关掉强类型检测，会把数字型转换为字符型，然后再进行比较。一般不推荐关闭强类型检测,这样可能会引发一些隐蔽的错误不利于程序稳定。

当两个数值不相等时，！=操作符将返回true,否则为false

```
trace(5 != 6);       // : true
trace(6 != 6);       // : false
trace(6 != "6");     // : false
trace(5 != "6");     // : true
```

同样，只有在关闭强类型检测后才能编译通过。
平时要注意不要把==写成=，否则会出现无法预料的错误。比如：

```
var quantity:int = 5;
// 下面的代码是错误的，正确应为 if (quantity == 6)
if (quantity = 6) {
   trace("Rabbits are bunnies.");
}
trace("quantity is " + quantity);    // 输出: quantity is 6
```

可以使用 *is* 操作符来检测数据类型

```
var quantity:int = 5;
if (quantity is int) {
```

```
    trace("Yippee. It's an integer.");
}
```

然而有些数值是非法的。下面的代码中quantity 等于 NaN (一个表示无效数字的常数)

```
var quantity:Number = 15 - "rabbits";
```

NaN 虽然是无效的数值，但它的数据类型仍属于 *Number*，

```
trace(typeof quantity);     // 显示: "number"
```

所以，为了测试有个number变量类型不是数字，但又是合法的number,尝试下这么写：

```
var quantity:Number = 15 - "rabbits";
if (quantity is Number) {
    //看起来好像正确，实际上是错误的，因为quantity != NaN 结果被认为都是false
    if (quantity != NaN) {
        trace("Yippee. It's a number.");
    }
}
```

为了检测一个数字是不合法的，要使用指定的函数*isNaN()* 看下面的例子：

```
var quantity:Number = 15 - "rabbits";
if (isNaN(quantity)) {
    trace("Sorry, that is not a valid number.");
}
```

如果要检测相反条件，只要取反就可以，比如为了检测一个变量是个合法的number，使用*!isNAN()*,如下：

```
var quantity:Number = 15 - "rabbits";
if (!isNaN(quantity)) {
  trace ("That is a valid number.");
}
```

当然了你还可以使用<和>比较符号来比较两个值得大小。

```
trace(5 < 6);       // 显示: true
trace(5 > 5);       // 显示: false
```

还有<= 和>= 符号

```
trace(5 <= 6);      // 显示: true
trace(5 >= 5);      // 显示: true
```

ActionScript 数据类型的比较有两个情况。在ActionScript中，数据类型分为两类：基本类型 (*string*, *number*, and *Boolean*) 和复合类型(*object*, *sprite*, and *array*)。当比较基本类型时，是比较他们的值，下面的例子中 quantity 和 total 被认为是相等的因为他们包含相同的值6

```
var quantity:Number = 6;
var total:Number = 6;
trace (quantity == total);              // 显示: true
```

然而，当比较符合数据类型时是通过他们的“引用”来比较。当两个引用所指向的对象完全相同才被认为是相等的，而不仅仅是对象的内容相同。例如，两个数组包含相同的内容，但是他们却不相等：

```
// 用相同的内容创建两个数组
var arrayOne:Array = new Array("a", "b", "c");
var arrayTwo:Array = new Array("a", "b", "c");
trace(arrayOne == arrayTwo);                // 显示: false
```

只要当引用指向同一个object, array, 或 sprite 才相等. 例子：

```
// 创建一个简单的数组
var arrayOne:Array = new Array("a", "b", "c");
// 创建另一个变量指向同一个数组
var arrayTwo:Array = arrayOne;
trace(arrayOne == arrayTwo);                // 显示: true
```

1.9. 执行条件语句

问题

我要当满足某些条件时才执行一些命令

解决办法

使用 *if* 或 *switch* 语句

讨论

我们经常需要让代码去有选择性的执行，这时可以使用 ActionScript 中的条件语句 *if*, *switch*, 或 三元条件运算符 (? :).

条件语句允许我们做出逻辑判断，某种情况下应该做什么。if语句是最简单的判断语句，当我们遇到多个可能的情况要处理，这时用switch更好些。而三元条件运算符是把检测和赋值都放在一行中搞定，简化操作。

首先我们来看一下*if* 语句， *if* 语句以if关键字开头，接着跟一对括号，括号内为测试表达式，后面的大括号放入测试表达式成立时要执行的代码。

下面的代码检测 animalName 是否等于"turtle."

```
if (animalName == "turtle") {   // 如果相等 trace( ) 语句将被执行
   trace("Yay! 'Turtle' is the correct answer.");
}
```

另外还可以加上 *else* 子句来处理当测试表达式不满足时的情况，要注意的是要看到trace()输出的信息则必须让程序在debug模式下运行。我们把输出信息放到*showMessage()* 方法里，这样这个函数就可以被重用了

```
if (animalName == "turtle") {   // 条件为真则执行
   showMessage("Yay! 'Turtle' is the correct answer.");
}else {    // 条件为假
   showMessage("Sorry, you got the question wrong.");
}
```

还可以加入*else if* 子句，如果 *if* 条件为真则跳过 *else if* 子句，如果为假则继续判断 *else if* 字句是否为真

```
if (animalName == "turtle") {    // 条件为真则执行
   showMessage ("Yay! 'Turtle' is the correct answer.");
}else if (animalName == "dove") { //animalName == "dove"成立则执行
   showMessage ("Sorry, a dove is a bird, not a reptile.");
}
```

还可以包含更多的 *else if* 子句，然而这种情况，最好的办法就是采用 *switch* 语句代替，因为 *switch* 与 *if* 语句结构更加清晰和简洁。 statements are more legible and succinct than the comparable *if* statement. 但在某些特殊场合，用 *if* 可以达到优化性能的目的。

switch 语句包含三部分：

switch 关键字

每个*switch* 语句都以switch 关键字开始

测试表达式

测试表达式被括号包围，它的结果将决定执行哪段代码。

switch 语句主体

主体中一般包含多个*cases*子句或一个default 子句

Case 表达式

case表达式将和 *switch* 表达式进行比较，如果相等就执行当前case的主代码。.

Case 主体y

当所有的case表达式都不等于switch表达式，将执行default主体。

```
switch (testExpression) {
   case caseExpression:
      // case body
   case caseExpression:
      // case body
   default:
      // case body
}
```

例子：

```
var animalName:String = "dove";
switch (animalName) {
   case "turtle":
      trace("Yay! 'Turtle' is the correct answer.");
   case "dove":
      trace("Sorry, a dove is a bird, not a reptile.");
   default:
      trace("Sorry, try again.");
}
```

一般情况下，在每个case主体最后都会加上 *break* 语句，这样执行完就会直接退出switch语句。

```
var animalName:String = "dove";
// 现在第2个case主体将被执行
switch (animalName) {
   case "turtle":
      trace("Yay! 'Turtle' is the correct answer.");
      break;
   case "dove":
      trace("Sorry, a dove is a bird, not a reptile.");
      break;
   default:
      trace("Sorry, try again.");
}
```

当有多个匹配但是执行代码是一样的，这时可以这么写：

```
switch (animalName) {
   case "turtle":
   case "alligator":
   case "iguana":
      trace("Yay! You named a reptile.");
      break;
   case "dove":
   case "pigeon":
   case "cardinal":
      trace("Sorry, you specified a bird, not a reptile.");
      break;
   default:
      trace("Sorry, try again.");
}
```


ActionScript 还支持三元条件运算符 (? :), 它把条件测试和赋值语句都放在一行完成。总共有3个操作数，第一个是条件表达式，如果为真，则取第二个操作数为结果，否则去第三个操作数为结果。

varName = (*conditional expression*) ? *valueIfTrue* : *valueIfFalse*;

1.10. 执行复杂的条件语句

问题

我要在多个条件中做出决定

解决办法

可以使用逻辑运算符*AND* (&&), *OR* (||), 和 *NOT* (!) 来创建符合条件语句。

讨论

ActionScript中的很多语句都能包含条件表达式。包括 *if*, *while*, 和 *for* 语句,如果测试两个条件都成立可以使用逻辑运算符 *AND*, &&, (更多细节请看第14章)：

```
// 测试今天是否是3月17号
var current:Date = new Date(    );
if (current.getDate(    ) == 17 && current.getMonth(    ) == 3) {
   trace ("Happy Birthday, Bruce!");
}
```

加入些括号让结构更加清晰：

```
// Check if today is April 17th.
if ((current.getDate(    ) == 17) && (current.getMonth(    ) == 3)) {
   trace ("Happy Birthday, Bruce!");
}
```

这里使用了逻辑运算符*OR*, ||, 来测试是否其中有个条件成立：

```
// 测试是否是周末
if ((current.getDay(    ) == 0) || (current.getDay(    ) == 6) ) {
   trace ("Why are you working on a weekend?");
}
```

还可以使用 *NOT*, !, 来测试条件不是真的：

```
// 检测名字不是Bruce.
if (!(userName == "Bruce")) {
   trace ("This application knows only Bruce's birthday.");
}
```

上面的例子还可以这么写：

```
if (userName != "Bruce") {
```

```
    trace ("This application knows only Bruce's birthday.");
}
```

任何布尔值或能得出布尔结果的表达式都能作为测试表达式：

```
// 检测如果sprite 是可见的，则输出信息
if (_sprite.visible) {
    trace("The sprite is visible.");
}
```

NOT 运算符经常被用来检测是否是false：

```
// 检测如果 sprite 是不可见的，则输出信息：
if (!_sprite.visible) {
    trace("The sprite is invisible. Set it to visible before trying this action.");
}
```

NOT 经常和*OR* 一起用：

```
// 检测既不是Bruce 有不是 Joey.
if (!((userName == "Bruce") || (userName == "Joey"))) {
    trace ("Sorry, but only Bruce and Joey have access to this application.");
}
```

1.11. 某段时间重复执行一种操作

问题

我要在单帧里多次执行某个任务

解决办法

在单帧里使用循环语句多次执行某个任务，例如，使用 *for* 语句：

```
for (var i:int = 0; i < 10; i++) {
   // 显示i.的值
   TRace(i);
}
```

讨论

使用循环语句可以让你的代码更加简洁。容易阅读和维护。既可以用 *while* 也可以用 *for* 语句，但是一般for语句比较好用。两个循环语句都能达到相同结果，只是 *for* 语句对于大多数程序员来说更熟悉些。

原形：

```
for (initialization; test, update) {
   statement body
}
```

下面的例子输出0到999的数字：

```
for (var i:int = 0; i < 1000; i++) {
   trace(i);
}
trace ("That's the end.");
```

多个初始值或步进值可以用逗号分开，初始化多个变量var 关键字只需要使用一次，下面的例子展示了每次i增加1，j减小1，然后输出i和j：

```
for (var i:int = 0, j:int = 10; i < 10; i++, j--) {
   trace("i is " + i);
   trace("j is " + j);
}
```

for 语句还可以嵌套，看下面的例子：

```
for (var i:int = 1; i <= 3; i++) {
   for (var j:int = 1; j <= 2; j++) {
```

```
        trace(i + " X " + j + " = " + (i * j));
    }
}
```

1 X 1 = 1

1 X 2 = 2

2 X 1 = 2

2 X 2 = 4

3 X 1 = 3

3 X 2 = 6

进行多重嵌套的for语句：

```
for (var i:int = 1; i <= 3; i++) {
    for (var j:int = 1; j <= 3; j++) {
        for (var k:int = 1; k <= 3; k++) {
            trace(i + " X " + j + " X " + k + " = " + (i * j * k));
        }
    }
}
```

许多开发都错误的用 *for* 语句让sprites运动起来; 比如：

```
for (var i:int = 0; i < 20; i++) {
    _sprite.x += 10;
}
```

上面的代码让sprite 向右移动200 像素，所有的更新都在同一帧完成，会出现两个问题：第一，场景每帧更新一次，所以只有最后的更新显示在场景中（导致我们看到好像是直接跳过200像素，而不是20步内慢慢移动过去）。第二，即使场景不停更新，但是for循环只需要几位秒，这样的动画也太快了。因此正确的做法是把动画放到 *enterFrame* 事件上执行。

再者若循环的代码执行时间超过15秒，Flash播放器就会提示警告。

1.12. 长时间执行一个任务

问题

我要长时间执行一个任务

解决办法

使用*Timer*类，或者监听sprite的*enterFrame*事件

讨论

Timer 类是ActionScript 3.0新增的，来代替早期的 *setInterval()* 和 *setTimeout()* 函数。当创建*Timer*类的实例时，它会在每个时间间隔激活 *timer* 事件，你可以在事件之间指定延时，然后就有足够的时间去激活 *Timer* 构造器了：

```
var timer:Timer = new Timer(delay, repeatCount);
```

使用 *addEventListener* 来设置一个函数处理这个事件，然后使用timer的 *start()* 方法启动或 *stop()* 停止它。

Timer 类属于 *flash.utils* 包，还有 *TimerEvent* 类在 *flash.events* 包中，因此需要导入它们：

```
package {
    import flash.display.Sprite;
    import flash.events.TimerEvent;
    import flash.utils.Timer;
    public class ExampleApplication extends Sprite {
        private var _PreviousTime:Number = 0;
        public function ExampleApplication(    ) {
            var tTimer:Timer = new Timer(500, 10);
            tTimer.addEventListener(TimerEvent.TIMER, onTimer);
            tTimer.start(    );
        }
        private function onTimer(event:TimerEvent):void {
            trace(flash.utils.getTimer(    ) - _PreviousTime);
            _PreviousTime = flash.utils.getTimer(    );
        }
    }
}
```

getTimer() 函数已经被移动到*flash.utils* 包中了。它返回程序开始有到现在的相对时间（微妙）

上个例子中，事件每隔5毫秒激活一次。如果你想模拟*setInterval()* 函数，把重复次数设为0。*stop()* 方法类似于*clearInterval()* 函数，停止定时器.

如果想模拟*setTimeout()*函数，设置重复数为1，定时器等到指定时间激活一次事件，然后停止。

Timer类最好的用处就是创建动画而不依赖于影片帧速。看下面的例子，两个定时器时间间隔分别为50微妙和100微妙：

```
package {
    import flash.display.Sprite;
    import flash.events.TimerEvent;
    import flash.utils.Timer;
    public class ExampleApplication extends Sprite {
        private var _square:Sprite;
        private var _circle:Sprite;
        public function ExampleApplication(  ) {
            // 创建两个图形
            _square = new Sprite(  );
            _square.graphics.beginFill(0xff0000);
            _square.graphics.drawRect(0, 0, 100, 100);
            _square.graphics.endFill(  );
            addChild(_square);
            _square.x = 100;
            _square.y = 50;
            _circle = new Sprite(  );
            _circle.graphics.beginFill(0x0000ff);
            _circle.graphics.drawCircle(50, 50, 50);
            _circle.graphics.endFill(  );
            addChild(_circle);
            _circle.x = 100;
            _circle.y = 200;
            // 创建两个定时器，启动
            var squareTimer:Timer = new Timer(50, 0);
            squareTimer.addEventListener(TimerEvent.TIMER, onSquareTimer);
            squareTimer.start(  );
```

```
            var circleTimer:Timer = new Timer(100, 0);
            circleTimer.addEventListener(TimerEvent.TIMER, onCircleTimer);
            circleTimer.start(    );
        }
        // 定义两个事件句柄
        private function onSquareTimer(event:TimerEvent):void {
            _square.x++;
        }
        private function onCircleTimer(event:TimerEvent):void {
            _circle.x++;
        }
    }
}
```

当然用 *enterFrame* 事件也可以实现的，但 *Timer* 技术更加灵活。

1.13. 创建可重用代码

问题

我要实现代码重用，而不是每次都去复制同样的代码。

解决办法

创建一个方法，然后再需要的地方调用它，类中的*函数* 我们通常称为*方法*。

怎样创建类方法：

控制修饰符 function *方法名* ( ):*返回数据类型* {

// 代码块

}

调用该方法只要饮用方法名就可以了，比如：

方法名( );

讨论

方法中的代码可以被多次执行。当你需要在不同的时间不同的地方执行同一个任务时就会很有用。把代码放在方法既便于理解又便于维护，而不用再多个地方修改。

像类变量一样，方法也有访问控制符。修饰符有：

private

只能被自身类访问。

protected

能被自身类实例或子类实例访问，其他类实例不能访问。

internal

可以被所在包中的所有类实例访问。

public

可被任何类访问。

如果都没指定修饰符，默认为internal。下面的代码定义了一个画线方法，然后被调用10次。

```
package {
    import flash.display.Sprite;
    public class ExampleApplication extends Sprite
    {
        public function ExampleApplication( ) {
            for(var i:int=0;i<10;i++) {
                drawLine( );
```

```
            }
        }
        private function drawLine(    ):void {
            graphics.lineStyle(1, Math.random(    ) * 0xffffff, 1);
            graphics.moveTo(Math.random(    ) * 400, Math.random(    ) * 400);
            graphics.lineTo(Math.random(    ) * 400, Math.random(    ) * 400);
        }
    }
}
```

还有种种要的方法类型是静态方法，静态方法不属于类实例，可以通过类直接调用静态方法。比如，有个类叫ExampleApplication, 定义了静态方法：

```
public static function showMessage( ):void {
    trace("Hello world");
}
```

可以这样调用：

```
ExampleApplication.showMessage(    );
```

有些类只有静态方法，Math 类就是个例子，注意我们使用 Math 方法时并没有创建类实例，我们只是调用了类属性那样调用类方法，比如 *Math.random()*, *Math.round()*, 等等

1.14. 增强代码可重用能力

问题

每次执行的任务都有微小的变化，但我又不想每次都复制那些代码修改一次。

解决办法

给方法传递参数让它适应不同的情况。

```
private function average (a:Number, b:Number, c:Number):void {
   trace("The average is " + (c + b + c)/3);
}
```

讨论

比如你有个求一系列数的平均数函数*average()*，你就可以把这些数字作为参数传递给函数去计算，而不必每次都去重写*average()函数*。

通常把参数都列在申明函数的括号内，多个参数用逗号分开。
下面有个简单的带有参数的函数申明：

```
//定义函数，带有两个参数： a 和 b.
private function average(a:Number, b:Number):Number {
   return (a + b)/2;
}//当函数被调用时，参数被传递进来,比如 5 和 11, 被传递给了 a 和 b
var averageValue:Number = average(5, 11);
```

大多数情况下，方法参数的个数可以预料的，但是有些情况下参数的个数是事先不确定的。比如：如果你想要让*average()* 方法接受任何数量的值，这时可以使用内建的数组，所有的参数都被放入函数数组中。

```
// arguments 数组
private function average(    ):Number {
   var sum:Number = 0;
   for (var i:int = 0; i < arguments.length; i++) {
      sum += arguments[i];
   }
   return sum/arguments.length;
} // 像下面这样传递任意数量的参数：
var average:Number = average (1, 2, 5, 10, 8, 20);
```

arguments 是一个 array 对象。

1.15. 从方法中退出

问题

我要从方法中退出

解决办法

方法中的代码被执行完就会自动退出，也可使用*return*语句直接退出。

讨论

return 语句将导致方法立即退出， ActionScript 解释器继续执行调用方法的所在位置的下面代码。方法中return下面的语句将被忽略。

```
private function sampleFunction (    ):void {
   return;
   trace("Never called");
}
```

下面的代码展示如果密码是错误的，则从方法中退出：

```
private function checkPassword (password:String):void {
   // 如果密码不是"SimonSays", 退出方法
   if (password != "SimonSays") {
      return;
   }
   //否则执行剩余的代码
   showForm ("TreasureMap");
}
// 使用错误的密码调用函数，所以函数退出
checkPassword("MotherMayI");
//使用正确的密码，所以显示“TreasureMap”信息.
checkPassword("SimonSays");
```

你可能注意到上面的例子方法被申明为void, 如果用return语句只是简单的退出这时可以的，但如果想返回值得话编译器就会报错了，如：

```
private function sampleMethod (    ):void {
   return "some value";    // This causes the compiler to generate an error.
}
```

1.16. 获得方法的执行结果

问题

我想执行一些方法，然后返回结果给调用它的函数

解决办法

使用 *return* 语句返回结果

讨论

return 返回的数据类型必须与函数申明的返回类型相一致。

```
private function average (a:Number, b:Number):Number {
   return (a + b)/2;
}
```

现在我们调用 *average()* 方法 然后把返回结果存到变量中，然后使用这个变量：

```
var playerScore:Number = average(6, 10);
trace("The player's average score is " + playerScore);
```

也可以不通过变量：

```
trace("The player's average score is " + average(6, 10));
```

注意，如果你不处理返回的值，那么返回结果就是丢失掉：

```
average(6, 10);
```

1.17. 处理异常

问题

我想让程序自己检测和处理遇到的异常。

解决办法

当检测到错误时使用 *throw* 语句抛出异常。把可能出现错误的代码都放到 *try* 块中，然后在 *catch* 块中进行错误处理。

讨论

Flash 播放器 8.5 开始支持 *try/catch 方法*来处理错误。这意味着可以灵活的处理遇到的错误了。除了语法错误（这时编译器就通不过），其他类型的错误如非法数据等都可以自己处理。

处理异常包括两个部分，抛出异常和捕获异常。有些异常系统会自动抛出，比如 *IllegalOperationError*, *MemoryError*, 和 *ScriptTimeoutError*. 它们都在 *flash.errors* 包中。除了系统定义的错误外也可以抛出自定义错误，然后捕获它进行处理。使用 *throw* 语句抛出一个*Error* 对象或*Error* 子类实例，比如：

```
throw new Error("A general error occurred.");
```

正如我们看到的，*Error* 构造器接受一个参数，这个信息和这个错误相关联。这个参数是可选的，依赖于你怎样处理这个错误，你可以不使用，但是大多数情况下都指定一个错误信息作为调试目的。

一旦异常被抛出，Flash就会暂停当前进程去寻找 *catch* 块去处理异常。任何有潜在错误的代码都要放在 *try* 块中，如果异常抛出，只有它所在的 *try* 块被暂停，然后相关联的 *catch* 块被调用，看下面的例子：

```
try {
   trace("This code is about to throw an error.");
   throw new Error("A general error occurred.");
   trace("This line won't run");
}
catch (errObject:Error) {
   trace("The catch block has been called.");
   trace("The message is: " + errObject.message);
}
```

上面的代码数出以下信息：

```
This code is about to throw an error.
The catch block has been called.
```

The message is: A general error occurred.

当然，上面的代码还是过于简单，但是这是个基本框架，可以看到只要抛出异常，*try* 块就会退出，*catch* 块被执行，传递了一个 *Error* 对象给 *catch*.

更多情况下，异常是从函数或方法中抛出的，Flash 会检测该函数是否在 *try* 块内被调用，如果是，则调用相应的 *catch* 块。

```
private function displayMessage(message:String):void {
    if(message == undefined) {
        throw new Error("No message was defined.");
    }
    trace(message);
}
try {
    trace("This code is about to throw an error.");
    displayMessage(    );
    trace("This line won't run");
}
catch (errObject:Error) {
    trace("The catch block has been called.");
    trace("The message is: " + errObject.message);
}
```

上面的代码输出以下内容：

This code is about to throw an error.

The catch block has been called.

The message is: No message was defined.

如果你不肯定你的函数或方法会在何时或如何抛出异常，这时就应该在try块进行调用。

```
// 定一个在指定的sprite里的画矩形函数。
private function drawRectangle(sprite:Sprite, newWidth:Number, newHeight:Number):void {
    // 检测长和宽的数值是否合法，否则抛出异常。
    if(isNaN(newWidth) || isNaN(newHeight)) {
        throw new Error("Invalid dimensions specified.");
    }
    // 如无异常，则画出矩形
```

```
    sprite.graphics.lineStyle(1, 0, 1);
    sprite.graphics.lineTo(nWidth, 0);
    sprite.graphics.lineTo(nWidth, nHeight);
    sprite.graphics.lineTo(0, nHeight);
    sprite.graphics.lineTo(0, 0);
}
```

现在我们在 try/catch 语句内调用该函数。

```
try {
    drawRectangle(this, widthB, heightB);
}
catch(errObject:Error) {
    this.graphics.clear(    );
    tOutput.text = "An error occurred: " + errObject.message;
}
```

另外对于*try/catch* 语句，还可以加入 *finally* 块，*finally* 块包含的代码无论是否遇到异常都会被执行。例如下面的两个例子效果相同：

```
//没有使用finally:
private function displayMessage(message:String):void {
    try {
        if(message == undefined) {
            throw new Error("The message is undefined.");
        }
        trace(message);
    }
    catch (errObject:Error) {
        trace(errObject.message);
    }
    trace("This is the last line displayed.");
}
//使用  finally:
private function displayMessage(message:String):void {
    try {
```

```
        if(message == undefined) {
            throw new Error("The message is undefined.");
        }
        trace(message);
    }
    catch (errObject:Error) {
        trace(errObject.message);
    }
    finally {
        trace("This is the last line displayed.");
    }
}
```

如果在catch中使用了return语句，那结果就不一样了：

```
//没有使用finally:
private function displayMessage(message:String):void {
    try {
        if(message == undefined) {
            throw new Error("The message is undefined.");
        }
        trace(message);
    }
    catch (errObject:Error) {
        trace(errObject.message);
        return;
    }
    // 这一句没有执行.
    trace("This is the last line displayed.");
}
//使用 finally:
private function displayMessage(message:String):void {
    try {
```

```
        if(message == undefined) {
            throw new Error("The message is undefined.");
        }
        trace(message);
    }
    catch (errObject:Error) {
        trace(errObject.message);
        return;
    }
    finally {
        // 执行，不管是否有异常发生。
        trace("This is the last line displayed.");
    }
}
```

通过这一节的学习，现在你可以建立复杂的异常处理系统。

2.0. 简介

ActionScript 3.0 最本质的东西就是类，也就说它是面向对象的。 ActionScript 3.0 在面向对象基础上重新构建了 ActionScript 核心。如果在 Flex 上编写 ActionScript 3.0，代码都被放在 <mx:Script> 标签内，所有 ActionScript 都必须以类的形式出现。这一章讨论在 ActionScript 3.0 上编写自定义类。

2.1. 创建自定义类

问题

我要编写自己的类

解决办法

保存一个以.as扩展名的新文件，类名和文件名相同，编写如下结构：

```
package package   {
    public class Class {
    }
}
```

讨论

在ActionScript 3 中，类是最基本的编程结构，所以必须先掌握编写类的基础知识。对于初学者，所以得类都必须放在.as文件中，每个as文件里只能定义一个public 类，而且类名字要与文件名相同。比如：你的类名为 *Example* ，那么文件名必须为 *Example.as*.

在 ActionScript 3.0 中所有的类都必须放在包中。包是对类进行分类的单位，其意义相当于文件系统的目录。包路径相对于classpath（类路径）,默认的类路径就是项目的根目录（就是包含mxml文件的所在目录），因此顶级的包目录就是项目根目录。包申明如下：

```
package name {
}
```

如果类定义在顶级包中，那么包名可以不指定，如：

```
package {
}
```

当类文件保存在子目录，那么包名就是它的保存目录，例如，文件保存在*example*目录，那么包这样申明：

```
package example {
}
```

如果类文件保存在 *example* 目录的子目录 *subpackage*, 应这样申明:

```
package example.subpackage {
}
```

包是很重要的，它可以避免类名称空间冲突。例如，有两个开发者写了两个类文件都叫 *MessageManager*. 这两个类虽有相同名字，但是完成不同的任务，因此你不能把这两个类放在一起，如果这样做，编译器将不知道调用哪个，一个办法是取个唯一的类名字。

你可以取名字叫 *EmailManager* 和 *BinarySocket- MessageManager*, 这是可以的，但是如果你管

理成千上万的类这时就很困难了。因此用包可以很好的解决这个问题，即使你有很多相同的类名，只要它们不在同一个包就不会冲突，如把 *MessageManager* 放在*net.messaging.email* 包另一个放在*net.messaging.binarysocket* 包中。

一般取包名都以自己的网站域名，这样可以最大限度避免和别人的包名相冲突。
当有多个项目公用一些类，那么这些类直接被放在主包 中的子目录中。例如，上面的 *MessageManager* 类 放 在 *com.examplecorp.net.messaging.email* 和 *com.examplecorp.net.messaging.binary- socket* 包中。

下一步就是申明类自身：

```
public class Name {
}
```

类申明必须在包内。下面的代码在顶级包中定义了叫 *Example* 的类：

```
package {
    public class Example {
    }
}
```

类主体在括号内定义，包括属性，方法。属性就是和类关联的变量，使用var关键字申明他们，属性也有修饰符指定其范围。修饰符有：
private

该属性只有类实例自身可访问.

public
该属性可以被任何类实例访问（若直接被类访问可设置成static）
protected

该属性只被自身类实例或派生类实例访问。

internal

该属性可被包内的类实例访问。

默认情况下是属性被指定为 internal ，除非自己指定修饰符。大多数情况，属性被指定为 private 或protected。按照习惯约定，private 和 protected 申明的属性名称都在前面加上下划线。看下面的例子：

```
package {
    public class Example {
        private var _id:String;
    }
}
```

与类关联的还有方法，你可以使用function关键字像申明函数那样申明方法。和属性一样，方法

也有修饰符（public, private, protected, internal）。如果方法被类实例访问可设置为public（直接被类访问则加上static）.如果方法只在类内方法则被设置为 private 或 protected。下面的代码申明一个方法叫*getId()*：

```
package {
    public class Example {
        private var _id:String;
        public function getId(  ):String {
            return _id;
        }
    }
}
```

按照约定，方法名称的起始字符必须为小写。每个类都有个和自己类名相同的方法，该方法称为构造函数，用它为创建新的实例时进行初始化工作。在 ActionScript 3.0 中，所有的构造函数都是public ,不像标准的方法，构造函数不能有返回值，也不能申明有返回类型。下面的代码申明了构造函数：

```
package {
    public class Example {
        private var _id:String;
        public function Example(  ) {
            _id = "Example Class";
        }
        public function getId(  ):String {
            return _id;
        }
    }
}
```

下面展示如何构造一个新的 *Example* 类实例：

```
var example:Example = new Example(  );
trace(example.getId(  )); // 显示： Example Class
```

2.2. 类的保存

问题

把类文件保存到哪里呢

解决办法

保存与包名称相符的目录中。

讨论

类文件保存在与包路径相符的目录中，比如：*com.examplecorp.net.messaging.email.MessageManager* 必须保存在 *com/examplecorp/net/messaging/email/目录下*. 编译器就知道去哪里找类。还有编译器也必须知道根目录是什么。例如，编译器需要知道 *com* 目录在哪里，编译器是通过classpath来找到com目录。默认的classpath就是Flex或Flash项目的根目录。例如，如果*com* 目录和.fla文件或mxml文件的同一目录，编译器就能找到这些类。其实你也可以保存到其他目录，例如，如果你有个公共库被多个项目使用，难道要拷贝多份到每个项目中，其实你可以编辑项目中的classpath来加入该库，这样不需要拷贝就能找到你的自定义类了。

右键点击工程名，选择Properties，选择Build Path，在Source Path中添加和修改classpath就可以了。如果你只使用SDK，那么当编译项目时必须设置classpath。使用*mxmlc* (Flex SDK中包含的命令行编译器), 加上 *-source-path 选项*， 跟上类目录，例如：

```
mxmlc -source-path . C:\libraries ExampleApplication.as
```

2.3. 创建成员属性

问题

我要创建public 成员属性

解决办法

使用隐含 getters 和 setters.

讨论

正如2.1节中所说的那样属性应该被申明为 private 或 protected。 public 属性并不是什么好主意，因为他不能体现封装性。要尽量做好封装，这意味着类不能暴露他的内部细节，public 属性使开发者能轻易破坏类或类实例。下面的简单例子是用了Public 属性：

```
package {
    public class Counter {
        public var count:uint;
        public function Counter(  ) {
            count = 0;
        }
    }
}
```

构造一个*Counter实例*, 然后改变count 属性值，如：

```
var counter:Counter = new Counter(  );
counter.count++;
```

但是，如果count属性被规定不能超过100，那么外部修改很可能无法保证，这样破坏了这个规定，一个办法就是设置 getters 和 setters，如下：

```
package {
    public class Counter {
        private var _count:uint;
        public function Counter(  ) {
            _count = 0;
        }
        public function getCount(  ):uint {
            return _count;
        }
```

```
        public function setCount(value:uint):void {
            if(value < 100) {
                _count = value;
            }
            else {
                throw Error(    );
            }
        }
    }
}
```

另一个办法就是用隐式getters 和 setters. 隐式 getters 和 setters 就像申明方法那样，但是看起来又像属性getter 语法如下：

```
public function get name(    ):Datatype {
}
```

setter 语法：

```
public function set name(value:Datatype):void {
}
```

下面定义隐式 getter 和 setter 方法：

```
package {
    public class Counter {
        private var _count:uint;
        public function Counter(    ) {
            _count = 0;
        }
        public function get count(    ):uint {
            return _count;
        }
        public function set count(value:uint):void {
            if(value < 100) {
                _count = value;
            }
            else {
```

```
                    throw Error(    );
                }
            }
        }
}
counter.count = 5;
trace(counter.count);
```

2.4. 创建静态方法或属性

问题

我要创建的方法和属性不需要类实例就能直接访问。

解决办法

使用static修饰符申明属性或方法

讨论

默认下属性和方法是属于实例的，例如 *Example* 类定义了 _id 属性和 *getId()* 方法，那么每个*Example 实例*都有自己的_id 属性和*getId()* 方法。但是有种情况你希望属性或方法是和类相关联而不是类实例，也就说不管有多少个类实例，都只有一个公共属性或方法，这样的属性和方法称为静态属性和方法。

Flash 播放器的类中就有些这样的例子，比如 *Math* 类中定义了 *round()* 方法，*round()* 方法就是个静态方法，因此可以通过类直接访问:

```
trace(Math.round(1.2345));
```

Math 类包含全部是静态方法，但是类也可以同时含有静态方法和实例方法及属性。比如 *String* 类有多数实例属性和方法，然而*fromCharCode()* 方法是静态的，该方法返回字符码。

下面的代码申明了一个静态的私有的属性_example:

```
static private var _example:String;
```

修饰符的顺序没有关系，比如static private 和 private static 是一样的.
static 最重要的用处就是在单态模式下，即类只能创建一个实例，单态类有一个 private static 属性用来存储类实例，然后在一个 public static 方法中访问这个实例。

2.5. 创建子类

问题

我要创建派生类

解决办法

使用extends关键字继承已有的类

讨论

如果新建的类和已有的类拥有公共特性但是又比已有类有更多细节，这时去重写所有的代码还不如从已有类中继承公共特性再添加专有代码。这时这个新类就是已有类的一个子类。

使用extends关键继承超类：

public class *Subclass* extends *Superclass*

子类可以引用任何超类中的 public 或 protected 的属性和方法，private 属性和方法是不能够访问的。

继承性是非常强大的功能，但是另一方面如何正确的使用继承也很重要。在写子类之前你要确定新类和超类是否已经有子类关系，这里有两种基本关系类型：继承和组合。你经常要确定类之间是"is a" 关系还是"has a" 关系：

"Is a" 关系是继承关系。如一个应用程序管理着一个图书馆的藏书。
"Has a" 关系是组合关系。大多数类使用组合，而且比继承更有伸缩性（然而需要更多代码）。例如一本书不是一个作者，但是它有一个作者。

图书馆有不同类型的藏品包括书和DVDs。很明显书和DVDs有不同的类型数据。书包括页数和作者，而DVDs则有播放时间数，演员，导演等等，但是也有一些公共数据，比如所有图书馆都有用一些唯一的数字编号或命名规则来管理这些藏品. 而且每个种类都有标题或名称，这时你可以定义个类来归纳这些公共信息：

```
package org.examplelibrary.collection {
    public class LibraryItem {
        protected var _ddc:String;
        protected var _id:String;
        protected var _name:String;
        public function LibraryItem(    ) {}
        public function setDdc(value:String):void {
            _ddc = value; }
        public function getDdc(    ):String {
            return _ddc; }
```

```
        public function setId(value:String):void {
            _id = value; }
        public function getId(    ):String {
            return _id; }
        public function setName(value:String):void {
            _name = value; }
        public function getName(    ):String {
            return _name; }
    }
}
```

书和DVDs 都是*LibraryItem*的一个种类. 接下来很自然就是定义*Book* 类和 *DVD* 类，*继承自LibraryItem*，*Book* 类类似于：

```
package org.examplelibrary.collection {
    import org.examplelibrary.collection.LibraryItem;
    public class Book extends LibraryItem {
        private var _authors:Array;
        private var _pageCount:uint;
        public function Book(    ) {}
        public function setAuthors(value:Array):void {
            _authors = value; }
        public function getAuthors(    ):Array {
            return _authors; }
        public function setPageCount(value:uint):void {
            _pageCount = value; }
        public function getPageCount(    ):uint {
            return _pageCount;
        }
    }
}
```

默认下可以继承任何类，但是如果你不要这个类再被继承，可以添加final修饰符申明该类：

```
final public class Example
```

2.6. 覆盖超类方法

问题

我要对从父类继承过来的方法进行重新实现。

解决办法

父类的方法必须申明为public 或 protected。当申明子类的实现时使用 override 修饰符

讨论

通常子类继承父类的所有方法而不做任何修改，但有些情况，继承过来的方法需要重新申明，实现与父类的方法不同。这时就要覆盖方法，该方法必须加上override 修饰符。如下面的例子，首先定义一个父类*Superclass*：

```
package {
    public class Superclass {
        public function Superclass( ) {}
        public function toString( ):String {
            return "Superclass.toString( )";
        }
    }
}
```

下一步，定义*Subclass* 继承自*Superclass*:

```
package {
    public class Subclass extends Superclass {
        public function Subclass( ) {}
    }
}
```

默认情况下，*Subclass* 继承 *Superclass* 实现的*toString( )* 方法：

```
var example:Subclass = new Subclass( );
trace(example.toString( )); // 显示: Superclass.toString( )
```

如果你要 *toString( )* 方法返回不同的值，就要覆盖它，如下：

```
package {
    public class Subclass extends Superclass {
        public function Subclass( ) {}
        override public function toString( ):String {
```

```
            return "Subclass.toString(    )";
        }
    }
}
```

覆盖方法，则该方法必须与父类的方法完全相同，包括参数数量和类型及返回类型，如果不一致，编译器就会报错。

有时你要覆盖方法是为了实现完全不同的功能，但有时你只是要增加一些东西，这时你可以调用父类方法：

```
super.methodName(    );
```

2.7. 创建常量

问题

我要怎么申明常量

解决办法

和申明属性差不多，只是在前面多了const 关键字

讨论

常量的值一旦定义就不可改变，这在有时候是很有用的。比如你有个复合的值需要经常用到，这时就可以把它当作简单的标示直接引用。Math.PI 就是个常量，MouseEvent.MOUSE_UP也是常量，包含mouseUp值，设定这些常量可以减少出错，比如下面的代码错误你难以觉察出来：

```
// 下面的代码没错，但是mouseUp 被写成了 mousUp，因此该代码无效。
addEventListener("mousUp", onMouseUp);
```

但是使用常量，编译器就会给出错误提示：

```
// 导致编译器错误
addEventListener(MouseEvent.MOUS_UP, onMouseUp);
```

常量也可以有static和public修饰符，这时常量一申明就要赋值：

```
static public const EXAMPLE:String = "example";
```

按照约定，常量都要大写，这样可以和其他变量或属性区别开来。

2.8. 发送事件

问题

我要发送事件

解决办法

继承 *flash.events.EventDispatcher* 然后调用 *dispatchEvent()* 方法

讨论

事件在对象之间的通讯起到至关重要的作用，有了它才能开发出功能强大的系统。Flash Player 9, 的*flash.events.EventDispatcher*类有一套事件发送机制。所有的事件都继承自*EventDispatcher*(比如 *NetStream* and *Sprite*). 如果你要定义个类要发送事件也要继承*EventDispatcher*，*如:*

```
package {
    import flash.events.EventDispatcher;
    public class Example extends EventDispatcher {
    }
}
```

EventDispatcher 类有个公共方法 *addEventListener()* 和 *removeEventListener()*， 通过它的子类来注册事件监听。 *EventDispatcher* 还有个 protected 方法 *dispatchEvent()* 来发送事件。*dispatchEvent()* 方法至少需要 *flash.events.Event* 对象或 *Event* 的子类作为参数。

3.0. 简介

Flash Player 9 关于控制运行时环境提供了更多的信。The *flash.system.Capabilities* 类有许多静态方法返回关于播放器和计算机的信息，比如操作系统，语言，音频和视频。还有其他的类如 *flash.display.Stage* 和 *flash.system.Security* 控制其他一些元素如播放器右键菜单和设置对话框。*flash.display.Stage* 类也控制影片剪辑缩放和对齐。

Flash Player 9 可能是自 Flash Player 7 之后最具重要意义的一次升级。它比以往提供了更多能力控制上下文菜单。在 Flash Player 7 里，用 *ContextMenu 类*，可以控制右键菜单，可以添加或删除菜单项。

3.1. 检测播放器版本

问题

我要确定客户机上的Flash播放器版本

解决办法

可以使用Flash Player检测工具。(http://www.adobe.com/software/flashplayer/download/detection_kit).

讨论

检测客户机上的Flash版本是个多年以来的难题，有各种开发者提供的方法，一般有三种方法：

基于浏览器脚本检测
服务端检测
ActionScript 检测

第一种方法使用JavaScript 或 VBScript 检测Flash 播放器版本。但是很多脚本在不同的平台不同的浏览器上会有兼容性问题出现。

服务端检测也有局限性，如果你无权限创建服务端脚本，这就很困难了。

大多基于 ActionScript 的检测技术不能在ActionScript 3.0 上用了。ActionScript 3.0 有一套自己的检测客户端版本的方法，那就是*flash.system.Capabilities.version* 属性。但是它不能检测Flash Player 8.5 之前的版本。

还好Adobe 已经考虑到所有这些问题，推出了 Flash Player Detection Kit 来指导你用最好的办法检测播放器版本。

检测包里包含文档和各种解决办法，包括VBScript 和 JavaScript 例子；ActionScript 检测；还有服务端的 ColdFusion 和 PHP 脚本检测。

基于ActionScript 的检测是比较好的，它可以支持到Flash播放器4，它使用一个 Flash 4 *.swf*t来检测当前版本，你所要做的就是在脚本里设置最小的版本变量，如果当前版本高，它会调用指定的内容。

3.2. 检测操作系统

问题

我要知道客户端的操作系统。

解决办法

使用flash.system.Capabilities.os 属性

讨论

ActionScript 3.0中，flash.system.Capabilities.os 属性返回操作系统名称和版本字符串。值可能包括Windows XP, Windows 2000, Windows NT, Windows 98/Me, Windows 95, 和 Windows CE. 在苹果机上，字符串包括版本号，比如 Mac OS 9.2.1 或 Mac OS X 10.4.4.

你可能基于操作系统做一些特殊处理，比如，根据当前系统载入特定的图标，或只是记录下用户的操作系统来统计。

下面的代码展示检测操作系统：

```
var os:String = System.capabilities.os.substr(0, 3);
if (os == "Win") {
   // Windows-specific code goes here
} else if (os == "Mac") {
   // Mac-specific code goes here
} else {
   // Must be Unix or Linux
}
```

3.3. 检测播放器类型

问题

我想知道播放器类型.

解决办法

使用 flash.system.Capabilities.playerType 属性.

讨论

播放器的类型有：

浏览器插件形式存在于 Mozilla 或 Firefox
ActiveX 控件形式存在于Internet Explorer
独立播放器
外部播放器，它与Flash IDE进行交互。

这些都是*.swf* 运行的环境，如果你要使用脚本进行交互，这就需要知道应用程序到底在Internet Explorer 或其他的浏览器运行。如果在独立播放器里运行，那么JavaScript等脚本就不管用了。

检测播放器类型，察看 flash.system.Capabilities.playerType 的值。它可能是PlugIn, ActiveX, StandAlone, 和 External：

```
if(flash.system.Capabilities.playerType == "Plugin") {
   // do actions for Mozilla, etc. browsers
}
else if(flash.system.Capabilities.playerType == "ActiveX") {
   // do actions for IE
}
else {
   // do actions for no browser
}
```

3.4. 检测系统语言

问题

我想知道客户端系统使用什么语言和输入法

解决办法

使用 flash.system.Capabilities.language 属性和 flash.system.IME 类

讨论

flash.system.Capabilities.language 属性给出客户端系统的语言，返回两个 ISO-639-1 字符（如 "fr" 代表 French). 有些国家代码两个字符是不合适的，比如("zh-CN" 代表 Simplified Chinese 和 "zh-TW" 代表 Traditional Chinese).

下面的代码展示如何使用语言属性：

```
// Example output: en-US
trace(flash.system.Capabilities.language);
var greetings:Array = new Array(    );
greetings["en"] = "Hello";
greetings["es"] = "Hola";
greetings["fr"] = "Bonjour";
var lang:String = flash.system.Capabilities.language.substr(0, 2);
if (greetings[lang] == undefined) {
   lang = "en";
}
trace(greetings[lang]);
```


如果要创建国际化的Flash，可以把文本保存在数组里，根据语言动态显示，或者直接做成多个Flash版本（每个语言一个），如 *myMovie_en.swf, myMovie_es.swf, myMovie_fr.swf,* 等.

```
//从 capabilities 对象上得到语言值
var lang:String = System.capabilities.language.substr(0, 2);
// 创建支持语言数组
var supportedLanguages:Array = ["en", "es", "fr"];
// 设置默认语言.
var useLang:String = "en";
//循环匹配，如果找到，设置 useLang
for (var i:int = 0; i < supportedLanguages.length; i++) {
```

```
    if (supportedLanguages[i] == lang) {
        useLang = lang;
        break;
    }
}
// 载入对应Flash
var movieURL:String =    "myMovie_" + useLang + ".swf");
```

还有一点也很重要，比如用户使用的输入语言，比如中文，日文，韩文，输入这些字符需要输入法，这时特定操作系统的一部分。

为了检测用户使用什么输入法， flash.system.Capabilities.hasIME 返回 true 或 false，flash.system.IME 类返回关于输入法的信息。flash.system.IME.enabled 属性设置用户是否可以使用输入法。在有些操作系统和版本上你可以发送字符串给 IME 来转换成正确的字符，接受 IME 的返回，但这不是所有操作系统都支持的，最好检测下先。

3.5. 检测显示设置

问题

我要知道客户机的显示设置情况

解决办法

使用 system.capabilities 对象的 screenResolutionX 和 screenResolutionY 属性

讨论

screenResolutionX 和 screenResolutionY 属性返回桌面的显示分辨率：

```
trace(flash.system.Capabilities.screenResolutionX);
trace(flash.system.Capabilities.screenResolutionY);
// 1024
// 768
```

有了这些值，你可以决定怎样显示flash影片。这一点对于Flash播放器也是很重要的，例如，手机屏幕和电脑屏幕尺寸是不同的，因此你要根据情况载入不同的尺寸的内容。

```
var resX:int = flash.system.Capabilities.screenResolutionX;
var resY:int = flash.system.Capabilities.screenResolutionY;
if ( (resX <= 240) && (resY <= 320) ) {
   var url:String = "main_pocketPC.swf";
}
else {
   var url:String = "main_desktop.swf";
}
loader.load(new URLRequest(url));
```

利用分辨率还可以居中你的弹出窗口：

```
var resX:int = flash.system.Capabilities.screenResolutionX;
var resY:int = flash.system.Capabilities.screenResolutionY;
//设置窗口的宽和高
var winW:int = 200;
var winH:int = 200;
//  设置窗口起始坐标
var winX:int = (resX / 2) - (winW / 2);
```

```
var winY:int = (resY / 2) - (winH / 2);
// 创建代码，然后传递给 URLLoader.load( )
// 打开新浏览器窗口
var jsCode:String = "javascript:void(
        newWin=window.open('http://www.person13.com/'," +
        "'newWindow', 'width=" + winW +
        ", height=" +  winH + "," +
        "left=" + winX + ",top=" + winY + "'));";
// 使用 URLLoader 对象调用 JavaScript 函数
urlLoader.load(new URLRequest(jsCode));
```

另外，得到屏幕分辨率可以确定是否缩放 Flash 影片。例如，用户把屏幕分辨率调高了，这时字体就变得小了。

3.6. 缩放影片

问题

我想让影片适应屏幕大小

解决办法

使用 stage.scaleMode 属性

讨论

这里有几种缩放模式：*exactFit*, *noBorder*, *noScale*, 和 *showAll*。为了避免编写上错误，这些字符串都成为了 flash.display.StageScaleMode 类的静态属性： EXACT_FIT, NO_BORDER, NO_SCALE, 和 SHOW_ALL.

Flash 播放器默认的缩放模式是*showAll*。这种模式会按照影片原始比例进行缩放以适应播放器大小。这样如果播放器的比例和影片的比例不一致就会导致电影边框的出现。设置应用程序的缩放模式：

```
stage.scaleMode = StageScaleMode.SHOW_ALL;
```

注意到 stage 并不是个全局对象，但是它是任何可视化对象的一个属性，因此这个语句在sprite类或继承自 DisplayObject 类里都可以。

noBorder 模式在保持原始比例下进行缩放以适应播放器，但是，如果播放器和影片比例不匹配，

影片显示不下的会被剪切掉，使用下面的语句设置：

stage.scaleMode = StageScaleMode.NO_BORDER;

exactFit 模式缩放影片适应播放器，它改变了电影原始比例，如果必要，它会匹配播放器，这样电影总是填充整个播放器，但是这样电影中的元素可能会扭曲，代码如下：

stage.scaleMode = StageScaleMode.EXACT_FIT;

noScale 模式即不进行缩放，保持原始比例。使用该模式不要忘了设置对齐方式（看 3.7 节）：

stage.scaleMode = StageScaleMode.NO_SCALE;

scaleMode 属性值并不影响右键菜单里功能，不过你可以禁用菜单里的缩放功能，看3.8 节。

3.7. 改变对齐方式

问题

我要改变影片的对齐方式

解决办法

使用 stage.align 属性

讨论

默认下 Flash 电影会居中显示。可以利用任何可视化对象的 stage.align 属性来重新设置电影的对齐方式。flash.display.StageAlign 类的属性：

Value	Vertical alignment	Horizontal
StageAlign.TOP	Top	Center
StageAlign.BOTTOM	Bottom	Center
StageAlign.LEFT	Center	Left
StageAlign.RIGHT	Center	Right
StageAlign.TOP_LEFT	Top	Left
StageAlign.TOP_RIGHT	Top	Right
StageAlign.BOTTOM_LEFT	Bottom	Left
StageAlign.BOTTOM_RIGHT	Bottom	Right

这里没有水平和垂直都居中的模式，其实，默认模式就是它了，但如果你改变了对其方式又想回到默认模式这时后只能传递空字符串""。

下面的类展示了缩放和对齐效果：

package {

```
    import flash.display.Sprite;
    import flash.display.StageScaleMode;
    import flash.display.StageAlign;
    public class ExampleApplication extends Sprite {
        public function ExampleApplication(    ) {
            stage.scaleMode = StageScaleMode.NO_SCALE;
            stage.align = StageAlign.TOP_RIGHT;
            graphics.beginFill(0xff0000);
            graphics.drawRect(0, 0, stage.stageWidth, stage.stageHeight);
            graphics.endFill(    );
        }
    }
}
```


3.8. 隐藏Flash播放器的菜单项

问题

我要隐藏右键菜单

解决办法

不能够完全改变Flash播放器的右键弹出菜单，但是可以设置stage.showDefaultContextMenu 属性为false来最小化菜单项。

讨论

默认下Flash播放器右键弹出菜单的项目有：

Zoom In
Zoom Out
Show All
Quality (Low, Medium, or High)
Settings
Print
Show Redraw Regions (debug 版本)
Debugger (debug 版本)

About Adobe Flash Player 9

通过下面的语句可以移除许多项目，Settings和About是不能移除的：

```
stage.showDefaultContextMenu = false;
```

遗憾的是，Flash 没有提供方法完全禁用右键菜单。

3.9. 检测设备音频

问题

我要确定播放器正在使用的音频设备.

解决办法

使用 flash.system.Capabilities 类的 hasAudio 和 hasMP3 属性

讨论

如果用户系统有播放音频的能力，则 flash.system.Capabilities.hasAudio 属性就返回True 。这实际上很重要，如果目标设备不支持音频，那就要避免强制用户下载音频内容（因此音频内容都比较大）。

```
// 只有当播放器可以播放声音才再如包含声音的.swf
if (flash.system.Capabilities.hasAudio) {
    content = "sound.swf";
} else {
    content = "silent.swf";
}
```

即时系统有播放音频能力也不意味着它有播放mp3的能力。因此当发布含有mp3内容时应用flash.system.Capabilities.hasMP3 属性检测下目标设备。

```
if (flash.system.Capabilities.hasMP3) {
    var url:URLRequest = new URLRequest("sound.mp3");
    sound = new Sound(url);
    sound.play(    );
} else {
    // code to load an external .swf containing a ADCP sound
}
```

3.10. 检测设备视频

问题

我要确定目标设备是否可以播放视频

解决办法

使用 flash.system.Capabilities 类的hasEmbeddedVideo, hasStreamingVideo, 和 hasVideoEncoder 属性

讨论

检测用户端是否能播放视频也同样重要，使用flash.system.Capabilities.hasStreamingVideo 属性检测是否能播放视频流。如果返回 false,就可以让用户下载内嵌视频的.swf文件，在这之前也要用flash.system.Capabilities.hasEbeddedVideo 确定，看下面的代码：

```
if(flash.system.Capabilities.hasStreamingVideo) {
   // 播放视频流
}
else if(flash.system.Capabilities.hasEmbeddedVideo) {
   // 下载内嵌视频的swf文件
}
else {
   //
}
```


如果应用程序需要视频流编码，比如传送摄像机视频流，还要确定系统是否具有编码能力，使用 flash.system.Capabilities.hasVideoEncoder 属性检测。

3.11. 提示用户改变播放器设置

问题

我要打开用户的Flash播放器设置对话框窗口

解决办法

使用 *flash.system.Security.showSettings()* 方法.

讨论

flash.system.Security.showSettings() 方法打开播放器设置对话框，它包含多个标签，也可以调用该方法时传递参数给它直接打开相应标签，该参数字符串是 *flash.system.SecurityPanel* 类的静态属性：

SecurityPanel.CAMERA

摄像机面板

SecurityPanel.DEFAULT

默认面板

SecurityPanel.LOCAL_STORAGE

本地存储面板

SecurityPanel.MICROPHONE

话筒面板

SecurityPanel.PRIVACY

安全控制面板

SecurityPanel.SETTINGS_MANAGER

设置管理面板会打开浏览器进行更多设置

如果没有传递参数，默认使用 *SecurityPanel.DEFAULT*. 下面的例子打开本地存储面板：

```
flash.system.Security.showSettings(SecurityPanel.LOCAL_STORAGE);
```

3.12. 处理系统安全

问题

我要在应用程序中载入其他域的swf文件，并且允许它访问程序中的 ActionScript

解决办法

使用*flash.system.Security.allowDomain()*,*flash.system.Security.allowInsecureDomain()*，或 一个策略文件。

讨论

很多情况下应用程序有多个分布在不同域里的swf组成。如果你要载入外部域的swf文件，需要通过 *flash.system.Security.allowDomain()*,*flash.system.Security.allowInsecureDomain()*, 或一个政策文件设定

假设*accessing.swf* 在*mydomain.com*，它要访问*otherdomain.com*中的*accessed.swf*中的一个变量，而默认*accessed.swf*是不允许外部域的swf访问它，为了解决这个问题，在*accessed.swf*中加入以下语句：

```
flash.system.Security.allowDomain("http://mydomain.com");
```

允许指定的域可以访问它。
也许你会注意到，被载入的swf如果要访问载入它的swf是不可以的，同样，载入它的swf也要加入上面的语句设置。
域名可以是字符串形式，也可以使IP地址。如果你想让所有域都能访问它，可以设置为 "*"。However, 但这样做可能会导致安全问题，不推荐。

如果 accessed *.swf* 文件在基于*https://*的服务器里，默认它不能被基于http://的域访问，设置*flash.system.Security.allowDomain( )* 也没用，这时应该使用*flash.system.Security.allowInsecureDomain()* 设置非安全的http域可以访问。

这个办法虽好，但是如果经常变动域名就要重新编译swf文件就麻烦了，最好的办法是创建一个策略文件.

该策略文件是一个 XML 文件，列出了被允许的域：

```
<?xml version="1.0"?>
<!-- http://www.mydomain.com/crossdomain.xml -->
<cross-domain-policy>
   <allow-access-from domain="www.otherdomain.com" />
   <allow-access-from domain="*.adobe.com" />
   <allow-access-from domain="123.45.67.89" />
</cross-domain-policy>
```

该文件被命名为 *crossdomain.xml*。通过 *flash.system.Security.loadPolicyFile()*读取文件，参数为

指定 *crossdomain.xml* 文件的URL字符串。

指定任何域都可访问:

<allow-access-from domain="*" />

阻止任何域访问:

<cross-domain-policy>

</cross-domain-policy>

4.0. 简介

数字的用法非常广泛，有多种表示形式，如十进制，十六进制，每种表示都有其特定用处，比如，十六进制经常来表示 RGB 颜色值。（看4.2关于如何在各种表示法之间进行转换） 。

数字和数学紧密相关，没有数学运算，Flash将非常迟钝。像加法和减法等简单运算在ActionScript经常用到，还有些运算，如随机数生成也是很常用的。

ActionScript 3.0 有三种基本数字类型： *number*, *int*, 和 *uint*。 number 对应浮点数， *int* 和 *uint* 对应整数。 *int* 和 *uint* 的区别就是 *uint* 代表无符号整数。

4.1. 数字的不同表现形式

问题

我要指定数字为十进制，八进制或十六进制

解决办法

十六进制以0x开头，八进制以0开头，二进制不能直接表示，可以用等价的八进制或十六进制或用 *parseInt()* 函数转换字符为数字。

讨论

ActionScript 中各种格式使用是很方便的。比如，如果你要设置Sprite.rotation 属性，最好是用十进制数：

```
rectangleSprite.rotation = 180;
```

另一方面，十六进制经常表示 RGB 颜色。例如，给 *ColorTransform* 对象的rgb属性使用十六进制数：

```
var pink:ColorTransform = new ColorTransform(    );

pink.rgb = 0xF612AB;
```

0X或0x开头的数字为十六进制数，十六进制有0到9和A到F字符组成，没有大小写之分。

0开头的数字为八进制，有0到7组成；例如,0777 是个八进制数。大多数开发者不怎么使用八进制，基本上都是使用十进制，除了颜色值用十六进制表示。

二进制只有 0 和 1 组成，虽然不能直接表示它，但你可以等效的十六进制来表示它，比如二进制 1111 等于十六进制 F，11111111 等于 FF，二进制数在为操作(&, |, ^, >>, <<, >>>)上经常用。.

4.2. 不同数字类型之间的转换

问题

我要把当前数字类型转换为别的数字类型

解决办法

用 *parseInt()* 函数把字符串转换为十进制数，用 *Number*, *uint*, 或 *int* 对象的 *toString()* 方法转换为字符串。

讨论

在ActionScript中不管你怎么设置数字，它的内部结果总是以十进制存贮：

```
// 创建颜色对象
var pink:ColorTransform = new ColorTransform(    );
// 用十六进制设置 RGB
pink.rgb = 0xF612AB;
// 显示这个值时：16126635
trace(pink.rgb);
```

如果你要输出为其他表示法，用 *toString(radix)* 方法

下面的例子用构造uint对象，输出不同的格式：

```
// radix 为 2, 输出二进制
trace(new uint(51).toString(2));    // 显示：  110011
// radix 为 16, 输出十六进制
trace(new uint(25).toString(16)); // 显示：19
var quantity:Number = 164;
trace(quantity.toString(16)); // 显示：  a4
```

下面的例子设置*ColorTransform* 对象的RGB值，调用 *toString()* 以十六进制显示：

```
var pink:Color = new ColorTransform(    );
pink.rgb = 0xF612AB;
trace(pink.rgb.toString(16));    // 显示：f612ab
```

toString() 方法的参数值的合法范围在2到36，如果没有指定参数值，默认为10。

和 *toString()* 相反的是 *parseInt()* 函数。它把指定的字符串转换为数字。

下面的代码把各种字符串，输出十进制数。

```
trace(parseInt("110011", 2));    // 显示: 51
```

```
trace(parseInt("19", 16));        // 显示: 25
trace(parseInt("17", 10));        // 显示: 17
```

如果不指定字符串进制，默认为十进制，除非在字符串前加上0x, 0X, 或0：

```
trace(parseInt("0x12"));          // 显示: 18
trace(parseInt("017"));           // 显示: 15
```

下面的例子给出的字符串格式和指定进制冲突，这时会默认为十进制

```
// 但是下面的字符串是不合法的数字，因此返回0
trace(parseInt("0x12", 10));      // 显示: 0
```

下面的字符串为八进制，但指定为十进制，因此系统默认字符串为十进制，而不是八进制。

```
trace(parseInt("017",    10));    // 显示 17
trace(parseInt("A9FC9C"));        // NaN
```

4.3. 四舍五入

问题

我要进行四舍五入或取近似值.

解决办法

用 *Math.round( )* 进行四舍五入， *Math.floor( )* 和 *Math.ceil( )* 进行上下近似值。 *NumberUtilities.round( )* 方法可自定义取值。

讨论

很多情况我们需要得到整数部分而不是带有小数的浮点数。比如计算出结果为 3.9999999 ，期望的结果应该是4.0。

Math.round( ) 方法进行四舍五入计算：

```
trace(Math.round(204.499));   // 显示: 204
trace(Math.round(401.5));     // 显示: 402
```

Math.floor( ) 方法去掉小数部分，*Math.ceil( )* 方法去掉小数部分后自动加1：

```
trace(Math.floor(204.99));    // 显示: 204
trace(Math.ceil(401.01));     // 显示: 402
```

如果我想要把90.337 四舍五入到 90.34,可以这么写：

```
trace (Math.round(90.337 / .01) * .01);    //显示: 9.34
trace (Math.round(92.5 / 5)  *  5);    // 显示: 95
trace (Math.round(92.5 / 10) * 10);   // 显示: 90
```

更好的办法是用自定义函数*NumberUtilities.round( )* ，它需要两个参数：
number ： 要舍入的数字
roundToInterval ： 间隔值

NumberUtilities 类在 *ascb.util* 包中。

imported *ascb.util.NumberUtilities*导入

```
trace(NumberUtilities.round(Math.PI));              // 显示: 3
trace(NumberUtilities.round(Math.PI, .01));         // 显示: 3.14
trace(NumberUtilities.round(Math.PI, .0001));       // 显示: 3.1416
trace(NumberUtilities.round(123.456, 1));           // 显示: 123
trace(NumberUtilities.round(123.456, 6));           // 显示: 126
trace(NumberUtilities.round(123.456, .01));         // 显示: 123.46
```

4.4. 格式化输出

问题

我要把数字进行格式化输出

解决办法

用 *NumberFormat* 类，设置掩码，然后调用 *format()* 方法。

讨论

经常会遇到需要在输出时在头部和尾部加0或空格来达到格式化输出的目的，比如显示时间或日期。比如要格式化输出6小时3分钟，显示为 6:03 或 06:03，而不是 6:3。而且还经常碰到输出时进行对齐等，这都需要进行格式化输出：

123456789

1234567

12345

虽然自己都可以做到这种效果，但是你会发现用 *NumberFormat* 对象更简单更灵活。*NumberFormat* 类是个自定义类，可到http://www.rightactionscript.com/ascb下载。使用它须先导入：import ascb.util.NumberFormat：
接下来决定是什么掩码进行格式化，它可以是0，#，.，,，或者其他。
零 (0)

使用零作为占位符

井号 (#)
点号(.)
逗号 (,)

下面看一下例子：掩码如下：

##,###.0000

格式化如下数字：1.2345, 12.345, 123.45, 1234.5, 和 12345， 结果：

1.2345

12.3450

123.4500

1,234.5000

12,345.0000

通过构造函数直接传入参数作为掩码：

```
var styler:NumberFormat = new NumberFormat("##,###.0000");
```

另外，使用属性进行修改掩码：

```
styler.mask = "##.00";
```

设置好掩码，调用 *format()* 方法就可以格式化数字了：

```
trace(styler.format(12345);
var styler:NumberFormat = new NumberFormat("#,###,###,###");
trace(styler.format(1));
trace(styler.format(12));
trace(styler.format(123));
trace(styler.format(1234));
styler.mask = "#,###,###,###.0000";
trace(styler.format(12345));
trace(styler.format(123456));
trace(styler.format(1234567));
trace(styler.format(12345678));
trace(styler.format(123456789));
```

输出如下：

```
                  1
                 12
                123
              1,234
             12,345.0000
            123,456.0000
          1,234,567.0000
         12,345,678.0000
        123,456,789.0000
```

NumberFormat 对象自动本地化返回值。如果Flash播放器在英文操作系统上运行，*NumberFormat* 类是用逗号和点号，如果在法语操作系统上运行，正好相反，使用点号和逗号，因此需要覆盖自动本地化：

有几种方法覆盖自动本地化设置：

把*Locale* 对象作为*format()* 方法的第二个参数，*format()* 方法根据*Locale* 对象的设置进行格式化。*Locale* 对象使用两个参数，第一个是语言代码，如en，第二个参数为国家代码，如EN.

通过全局设置*Locale.slanguage* 或 *Locale.svariant* 属性就不需要给*format()* 方法传递参数了

使用符号对象作为*format()*的第二个参数，包括两个属性：group和decimal。该方法适合没有

用Locale对象进行全局设置下使用。

Locale 类在 *ascb.util* 包中，使用前导入。

下面的代码展示了三种方法：

```
var styler:NumberFormat = new NumberFormat("#,###,###,###.00");
Locale.slanguage = "fr";
trace(styler.format(1234));
trace(styler.format(12345, {group: ",", decimal: "."}));
trace(styler.format(123456));
Locale.slanguage = "en";
trace(styler.format(1234567));
trace(styler.format(12345678, new Locale("es", "ES")));
trace(styler.format(123456789, {group: "|", decimal: ","}));
```

结果：

```
          1.234,00
         12,345.00
        123.456,00
      1,234,567.00
     12.345.678,00
   123|456|789,00
```

4.5. 不使用掩码进行数字格式化

问题

我不想使用掩码进行格式化

解决办法

用 *NumberFormat* 对象不设置掩码，调用 *format()* 方法

讨论

4.4 节讨论了各种复杂的数字格式化方法，但是能不能不用那么复杂呢，*NumberFormat* 类提供了一个简单的办法，只使用最简单的*format()*方法：

```
var styler:NumberFormat = new NumberFormat(    );
trace(styler.format(12.3));
trace(styler.format(123.4));
trace(styler.format(1234.5));
trace(styler.format(12345.6));
```

显示如下：

12.3

123.4

1,234.5

12,345.6

正像上一节所说的，仍然有本地化问题，本地化处理和上一节一样：

```
var styler:NumberFormat = new NumberFormat(    );
Locale.slanguage = "fr";
trace(styler.format(1234, new Locale("en")));
trace(styler.format(12345, {group: ":", decimal: "|"}));
trace(styler.format(123456));
```

输出：

1,234

12:345

123.456

4.6. 格式化货币数字

问题

我要格式化货币，比如美元

解决办法

使用 *NumberFormat.currencyFormat()* 方法

讨论

不像其他语言，比如ColdFusion，ActionScript 没有提供内建的函数格式化货币数字。自定义类 *NumberFormat* 包括一个*currencyFormat()*方法。

currencyFormat() 至少需要一个参数，看下面的简单代码：

```
var styler:NumberFormat = new NumberFormat(    );
trace(styler.currencyFormat(123456));
```

在英文操作系统上运行结果如下：

$123,456.00

和 *format()* 方法类似，*currencyFormat()* 方法根据自动本地化设置进行格式化。因此，如果上面的代码在西班牙语操作系统上运行结果如下：

123.456,00

覆盖自动本地化方法和format()类似：

使用 *Locale* 对象作为*currencyFormat()*的第二个参数.
赋全局变量给 *Locale.slanguage* 和 *Locale.svariant* 属性
使用字符对象作为*currencyFormat()*的第二个参数.

currencyFormat() 的字符对象比*format()* 方法中稍微不一样，它包括4个属性： group, decimal, currency, 和 before。group 和 decimal 属性和 *format()* 方法一样。currency 属性为货币符号，before 为布尔值，表示货币符号的位置。

下面是*currencyFormat()*的一些例子代码：

```
var styler:NumberFormat = new NumberFormat(    );
trace(styler.currencyFormat(123456));
Locale.slanguage = "nl";
trace(styler.currencyFormat(123456));
trace(styler.currencyFormat(123456, new Locale("sv")));
trace(styler.currencyFormat(123456, {group: ",", decimal: ".", currency: "@", before: false}));
```

输出结果：

$123,456.00

123.456,00

123,456.00kr

123,456.00@

4.7. 生成随机数

问题

我要生成随机数

解决办法

使用*Math.random()* 方法生成 0 到 .999999的随机数。还有，是用*NumberUtilities.random()* 方法可以生成指定范围的随机数

讨论

Math.random() 方法产生 0 到 0.999999999的浮点随机数。大多数情况我们希望产生整数而不是浮点数，还好随机值可以指定精度。

NumberUtilities.random() 方法产生指定的范围和精度，它接受三个参数：

minimum

可接受的最小值

maximum

可接受的最大值

roundToInterval

间隔值，可选

NumberUtilities 类在 *ascb.util* 包中，使用前导入。
例子如下：

```
// 产生 0 到 100的整数.
trace(NumberUtilities.random(0, 100));
// 产生 0 到 100的整数，间隔为5
trace(NumberUtilities.random(0, 100, 5));
trace(NumberUtilities.random(-10, 10, .1));
trace(NumberUtilities.random(-1, 1, .05));
package {
   import flash.display.Sprite;
```

```
    import ascb.util.NumberUtilities;
    import flash.utils.Timer;
    import flash.events.TimerEvent;
    public class RandomNumberTest extends Sprite {
        private var _total:uint;
        private var _numbers:Object
        public function RandomNumberTest(    ) {
            var timer:Timer = new Timer(10);
            timer.addEventListener(TimerEvent.TIMER, randomizer);
            timer.start(    );
            _total = 0;
            _numbers = new Object(    );
        }
        private function randomizer(event:TimerEvent):void {
            var randomNumber:Number = NumberUtilities.random(1, 10, 1);
            _total++;
            if(_numbers[randomNumber] == undefined) {
                _numbers[randomNumber] = 0;
            }
            _numbers[randomNumber]++;
            trace("random number: " + randomNumber);
            var item:String;
            for(item in _numbers) {
                trace("\t" + item + ": " + Math.round(100 * _numbers[item]/_total));
            }
        }

    }
}
```

4.8. 模拟硬币投掷

问题

我要模拟硬币投掷或布尔事件来达到50%几率成功。

解决办法

用*NumberUtilities.random()* 方法产生 0 到 1的整数，根据每种可能得出结果。

讨论

用 *random()* 方法产生指定范围的随机整数，能够产生两个结果对应硬币的正面和反面状态，在程序里我们用0代表一个状态，1代表另一状态，当然你用1和2也是可以的，总之是2个状态，这样就能模拟硬币投掷了：

```
package {
   import flash.display.Sprite;
   import flash.text.TextField;
   import flash.events.MouseEvent;
   import ascb.util.NumberUtilities;
   public class CoinExample extends Sprite {
      private var _field:TextField;
   public function CoinExample(    ) {
      _field = new TextField(    );
      _field.autoSize = "left";
      addChild(_field);
      var circle:Sprite = new Sprite(    );
      circle.graphics.beginFill(0, 100);
      circle.graphics.drawCircle(100, 100, 100);
      circle.graphics.endFill(    );
         circle.addEventListener(MouseEvent.CLICK, onClick);
         addChild(circle);
   }
   private function onClick(event:MouseEvent):void {
         var randomNumber:Number = NumberUtilities.random(0, 1);
         _field.text = (randomNumber == 0) ? "heads" : "tails";
      }
```

```
    }
}
```

记录正反面出现次数：

```
package {
    import flash.display.Sprite;
    import flash.text.TextField;
    import ascb.util.NumberUtilities;
    public class CoinTest extends Sprite {
        private var _field:TextField;
    public function CoinTest(    ) {
        _field = new TextField(    );
        _field.autoSize = "left";
        addChild(_field);
        var heads:Number = 0;
        var tails:Number = 0;
        var randomNumber:Number;
        for(var i:Number = 0; i < 10000; i++) {
            randomNumber = NumberUtilities.random(0, 1);
            if(randomNumber == 0) {
                heads++;
            }
            else {
                tails++;
            }
        }
        _field.text = "heads: " + heads + ", tails: " + tails;
    }
    }
}
```

如果要测试随机数，应该保存在变量中，像上面的代码，下面代码的统计是错误的，因此else if每次都产生新的随机数：

```
package {
```

```
    import flash.display.Sprite;
    import ascb.util.NumberUtilities;
    public class RandomLetter extends Sprite {
        public function RandomLetter(    ) {
            for(var i:Number = 0; i < 10000; i++) {
                trace(getRandomLetter(    ));
            }
        }
        private function getRandomLetter(    ):String {
            if(NumberUtilities.random(0, 2) == 0) {
                return "A";
            }
            else if(NumberUtilities.random(0, 2) == 1) {
                return "B";
            }
            else if(NumberUtilities.random(0, 2) == 2) {
                return "C";
            }
            // It's possible that none of the preceding will evaluate to true,
            // and the method will reach this point without returning a valid
            // string.
            return "";
        }
    }
}
```

下面才正确：

```
package {
    import flash.display.Sprite;
    import ascb.util.NumberUtilities;
    public class RandomLetter extends Sprite {
        public function RandomLetter(    ) {
```

```
            for(var i:uint = 0; i < 10000; i++) {
                trace(getRandomLetter(    ));
            }
        }
        private function getRandomLetter(    ):String {
            // random(    ) 返回结果到变量上
            var randomInteger:uint = NumberUtilities.random(0, 2);
            if(randomInteger == 0) {
                return "A";
            }
            else if(randomInteger == 1) {
                return "B";
            }
            else if(randomInteger == 2) {
                return "C";
            }
            return "";
        }
    }
}
```

4.9. 模拟骰子

问题

我要模仿掷骰子

解决办法

用*NumberUtilities.random()* 方法产生指定范围的随机数

讨论

用 *random()* 方法产生整数来模拟掷骰子，这在很多游戏中经常用到，这次我们在ActionScript中实现

一般我们产生随机数然后保存它在使用，如果要重新使用存在的随机数，应保存它而不是再产生新的随机数。注意下面两种情况，第一种，dice总是die1和die2的之和：

```
var die1:uint = NumberUtilities.random(1, 6);
var die2:uint = NumberUtilities.random(1, 6);
var dice:uint = die1 + die2;
```

下面的情况，dice和die1和die2没有关系，换句话说，即使die1和die2加起来等于7，dice也不会等于它：

```
var die1:uint = NumberUtilities.random(1, 6);
var die2:uint = NumberUtilities.random(1, 6);
var dice:uint = NumberUtilities.random(1, 6) + NumberUtilities.random(1, 6);
```

NumberUtilities.random() 还可以模拟多边的骰子：

```
var die1:uint = NumberUtilities.random(1, 15);
```

下面的例子模拟了一个骰子

```
package {
   import flash.display.Sprite;
   import flash.text.TextField;
   import flash.events.MouseEvent;
   import ascb.util.NumberUtilities;
   public class NumbersAndMath extends Sprite {
      var _die:Sprite;
      var _value:uint;
      public function NumbersAndMath(    ) {
         _die = new Sprite(    );
```

```
            addChild(_die);
            _die.addEventListener(MouseEvent.CLICK, rollDie);
            rollDie(null);
        }
        private function rollDie(event:MouseEvent):void {
            _value = NumberUtilities.random(1, 6);
            _die.graphics.clear(    );
            _die.graphics.lineStyle(    );
            _die.graphics.beginFill(0xFFFFFF);
            _die.graphics.drawRect(0, 0, 50, 50);
            _die.graphics.endFill(    );
            _die.graphics.beginFill(0x000000);
            if(_value == 1 || _value == 3 || _value == 5) {
                _die.graphics.drawCircle(25, 25, 4);
            }
            if(_value == 2 || _value == 3 || _value == 4 || _value == 5 || _value == 6)
            {
                _die.graphics.drawCircle(11, 11, 4);
                _die.graphics.drawCircle(39, 39, 4);
            }
            if(_value == 4 || _value == 5 || _value == 6) {
                _die.graphics.drawCircle(11, 39, 4);
                _die.graphics.drawCircle(39, 11, 4);
            }
            if(_value == 6) {
                _die.graphics.drawCircle(11, 25, 4);
                _die.graphics.drawCircle(39, 25, 4);
            }
        }
    }
}
```

4.10. 产生唯一的随机数

问题

我要产生唯一数

解决办法

使用 *NumberUtilities.getUnique()* 方法

讨论

唯一随机数经常在产生唯一的URL 时用到。就是在URL后加上个唯一的数字，以区别于使用过的URL，因此浏览器总是会去调用远程服务器而不是访问缓存

NumberUtilities.getUnique() 返回基于毫秒的数字

```
trace(NumberUtilities.getUnique(    ));
```

下面的代码产生一组唯一的随机数：

```
for(var i:Number = 0; i < 100; i++) {
   trace(NumberUtilities.getUnique(    ));
```


4.11. 转换角度计算

问题

我要计算角度及转换为合适的单位

解决办法

使用 *Unit* 和 *Converter* 类

讨论

影片剪辑的_rotation 属性使用角度计算的。如果用弧度而不是角度就有些麻烦了。首先要把弧度值转换为角度值，再赋值给 _rotation 属性，而且，大多数人喜欢用角度计算。还好，把弧度转换为角度挺容易，只要180/Math.PI.，角度转换为弧度就反一下，Math.PI/180. ，而且使用自定义的*Unit* 和*Converter* 类更简单。

这两个类都是自定义类，在*ascb.unit* 包中，首先创建*Unit* 实例，然后描述需要转换的单位类型，

Unit.DEGREE, Unit.RADIAN, 和 Unit.GRADIAN 常量返回新的Unit 对象表示各种单位。*Unit* 对象有些属性，包括name, category, label, 和labelPlural:

```
var degree:Unit = Unit.DEGREE;
trace(degree.name);              // 显示: degree
trace(degree.category);      // 显示: angle
trace(degree.label);             // 显示: degree
trace(degree.labelPlural); // 显示: degrees
```

使用*getConverterTo()* 方法，传递 *Unit* 对象作为，得到converter对象，看下面的代码得到弧度向角度转换的对象：

```
var converter:Converter = Unit.DEGREE.getConverterTo(Unit.RADIAN);
```

一旦得到Converter 实例，运行*convert()* 方法，指定值进行转换：

```
trace(converter.convert(90));
```

convertWithLabel() 方法输出字符串：

```
var converterToRadians:Converter = Unit.DEGREE.getConverterTo(Unit.RADIAN);
var converterToDegrees:Converter = Unit.RADIAN.getConverterTo(Unit.DEGREE);
trace(converterToRadians.convertWithLabel(1));
trace(converterToRadians.convertWithLabel(57.2957795130823));
trace(converterToDegrees.convertWithLabel(1));
trace(converterToDegrees.convertWithLabel(0.0174532925199433));
/*
    显示:
    0.0174532925199433 radians
    1 radian
    57.2957795130823 degrees
    1 degree
*/
```

如果执行相反操作，使用*getConverterFrom()* 方法：

```
var converter:Converter = Unit.DEGREE.getConverterFrom(Unit.GRADIAN);
trace(converter.convert(100));
trace(converter.convert(23));
```

4.12. 计算两点之间的距离

问题

我要计算两点之间的距离

解决办法

根据勾股定理，使用 *Math.pow()* 和 *Math.sqrt()* 联合计算

讨论

通过勾股定理可以计算出两点之间的距离（直线）。一个三角形，最长边的平方等于其他两边的平方和：

$$a^2 + b^2 = c^2$$

根据这个公式可以计算出两点之间的距离，*a* 是两点X坐标的差值，*b* 是两点Y坐标的差值：

```
var c:Number = Math.sqrt(Math.pow(a, 2) + Math.pow(b, 2));
```

5.0. 简介

数组保存一组相关联的数据，组织和处理这些数据。数组概念在生活中是很常见的，比如菜谱，包含姓名，地址，生日等地址本都是数组原理。

在ActionScript中，有两种数组：整型下标和联合数组，都是组织相关数据，只是访问数据的方法不同而已。

整型下标数组：

数组的每个元素都用唯一的整数下标来索引。通过下标排序，起始值为0，每个元素保存在索引指定的位置，这就像抽屉一样。

联合数组 ：

用字符串关键字作为每个元素的索引。

首先创建数组，有两种构造方法，

```
// 创建空的数组
var array:Array = new Array();
// 创建数组时指定数组长度
var array:Array = new Array(elements);
//创建数组时加入多个元素
var array:Array = new Array(element0,...elementN);
```

直接用数组符号也可以创建一个数组，这是很简洁的方式创建数组：

```
var letters:Array = ["a", "b", "c"];
```

Array 类提供了一些方法修改数组内容或者返回新的数组

使用数组操作符（中括号加上下标索引）来读取和设置内容，如：

```
//设置第5个元素为"apples"
// (下标从0开始).
items[4] = "apples";
// 显示第5个元素
trace(items[4]);     // 显示: apples
```

ActionScript 并不关心数组里存的是什么类型的数据，可以是字符串，数字，布尔值和引用的类型，而且不像其他语言，同一个数组可以存不同类型的数据，例如：

```
var data:Array = ["a", 2, true, new Object()];
```

还有点不同，数组在创建时可以不用指定数组长度。

5.1. 在数组首部和尾部添加元素

问题

我要再添加新元素到数组

解决办法

push() 方法在数组尾部添加元素，*unshift()* 方法在数组首部插入元素

讨论

Array.push() 方法把元素加在数组的尾部，也可以一次添加多个值：

```
var array:Array = new Array();
array.push("val 1", "val 2");
```

也可以在数组指定的下标位置设置值，下标位置在0到*Array* .length - 1之间：

```
array[array.length] = "val 3";
```

如果下标不存在，数组会自动扩充，扩充的空间当中，没有被赋值的自动以“undefined”填充：

```
var letters:Array = ["a", "b", "c"];
letters[5] = "f";
```

通过*unshift()* 方法在数组首部插入新元素：

```
// 创建四个元素的数组
// "a", "b", "c", "d".
var letters:Array = new Array(    );
letters.push("a", "b", "c", "d");
// 添加"z" ，其他元素依次向下移动
letters.unshift("z");
for (var i:int = 0; i < letters.length; i++) {
   trace(letters[i]);
}
```

到底把数据插入到哪里需要根据具体实际需要，比如要达到(LIFO) 目的，我们需要 *Array.push()* 和 *Array.pop()* 成对使用。

5.2. 遍历数组成员

问题

我要访问数组的每个元素

解决办法

利用 *for* 循环来遍历数组，使用下标返回元素。

讨论

for循环的初始变量从0开始，结束为array.length-1，因为是从下标0开始的：

```
var letters:Array = ["a", "b", "c"];
for (var i:int = 0; i < letters.length; i++) {
   trace("Element " + i + ": " + letters[i]);
}
```

也可以降序遍历数组，循环变量从array.length-1开始到0：

```
var letters:Array = ["a", "b", "c"];
for (var i:int = letters.length - 1; i >= 0; i--){
   trace("Element " + i + ": " + letters[i]);
}
```

有很多情况需要用循环遍历所有元素，比如，获得了包含sprite的数组，然后把每个sprite的x坐标+1：

```
for (var i:int = 0; i < sprites.length; i++){
   sprites[i].x++;
}
```

可以把数组长度存在变量中，免得每次循环都要重新计算：

```
var length:int = sprites.length;
for (var i:int = 0; i < length; i++){
   sprites[i].x++;
}
```

这样做可以提高 Flash 性能，因为不用每次循环都去计算长度了，但是有个前提，就是没有进行插入删除操作来改变长度值，否则就要每次计算长度才行。

5.3. 搜索匹配的数组元素

问题

我要找出指定值得数组元素

解决办法

用*for* 语句和*break* 语句就能找到匹配的元素。另外用*ArrayUtilities.findMatchIndex( )*, *ArrayUtilities.findLastMatchIndex( )*, 和*ArrayUtilities.findMatchIndices( )* 方法

讨论

用for循环查找第一个匹配的元素后，用break立即返回，这样就实现功能了。

break应该在if语句里进行判断，是否找到匹配元素，找到则执行break推出循环，否则继续查找。

```
var letters:Array = ["a", "b", "c", "d", "a", "b", "c", "d"];
// 指定要搜索的内容
var match:String = "b";
for (var i:int = 0; i < letters.length; i++) {
    // 检测当前元素是否匹配
    if (letters[i] == match) {
        trace("Element with index " + i +
                " found to match " + match);
        break;
    }
}
```


也可以找到匹配的最后一个元素，这就需要倒序遍历数组：

```
var letters:Array = ["a", "b", "c", "d", "a", "b", "c", "d"];
var match:String = "b";
for (var i:int = letters.length - 1; i >= 0; i--) {
    if (letters[i] == match) {
        trace("Element with index " + i +
                " found to match " + match);
        break;
    }
}
```

使用自定义类*ArrayUtilities* 类更简单，它在 *ascb.util* 包中，首先导入它：

```
import ascb.util.ArrayUtilities;
```

ArrayUtilities 类有三个方法来查找匹配的元素findMatchIndex( ), findLastMatchIndex( ), 和findMatchIndices( )。 findMatchIndex( ) 方法至少需要两个参数：一个指向数组的引用和需要匹配的值，返回第一个匹配的元素下标，如果找不到返回-1：

```
var letters:Array = ["a", "b", "c", "d"];
trace(ArrayUtilities.findMatchIndex(letters, "b"));
// 显示: 1
trace(ArrayUtilities.findMatchIndex(letters, "r"));
// 显示: -1
```

也可以指定搜索的起始下标作为第三个参数：

```
var letters:Array = ["a", "b", "c", "d", "a", "b", "c", "d"];
trace(ArrayUtilities.findMatchIndex(letters, "a", 1));
// 显示: 4
```

如果第三个参数为true，则返回部分匹配的元素：

```
var words:Array = ["bicycle", "baseball", "mat", "board"];
trace(ArrayUtilities.findMatchIndex(words, "s", true));
// 显示: 1
```

如果你想部分匹配又想指定起始搜索下标，可以把起始下标作为第四个参数。

findLastMatchIndex( ) 方法返回最后一个匹配的元素下标

findMatchIndices( ) 方法返回所有匹配的元素下标数组：

```
var letters:Array = ["a", "b", "c", "d", "a", "b", "c", "d"];
trace(ArrayUtilities.findMatchIndices(letters, "b"));
// 显示: 1,5
```

也可以设定为部分匹配，指定第三个参数为true：

```
var words:Array = ["bicycle", "baseball", "mat", "board"];
trace(ArrayUtilities.findMatchIndices(words, "b", true));
// 显示: 0,1,3
```

ArrayUtilities 方法内部也是用for循环来实现的，现在我们看看代码，下面是findMatchIndex( ) 方法的代码：

```
public static function findMatchIndex(array:Array, element:Object):int {
    // Use a variable to determine the index
```

```
// from which to start. Use a default value of 0.
var startingIndex:int = 0;
// By default don't allow a partial match.
var partialMatch:Boolean = false;
// If the third parameter is a number,
// assign it to nStartingIndex.
// Otherwise, if the fourth parameter is a number,
// assign it to nStartingIndex instead.
if(typeof arguments[2] == "number") {
        startingIndex = arguments[2];
}
else if(typeof arguments[3] == "number") {
        startingIndex = arguments[3];
}
// If the third parameter is a Boolean value,
// assign it to partialMatch.
if(typeof arguments[2] == "boolean") {
        partialMatch = arguments[2];
}
// Assume no match is found.
var match:Boolean = false;
// Loop through each of the elements of the array
// starting at the specified starting index.
for(var i:int = startingIndex;
                i < array.length; i++) {
        // Check to see if the element either matches
        // or partially matches.
        if(partialMatch) {
                match = (array[i].indexOf(element) != -1);
        }
        else {
```

```
                match = (array[i] == element);
            }
            // If the element matches, return the index.
            if(match) {
                return i;
            }
        }
        // The following return statement is only reached
        // if no match was found. In that case, return -1.
        return -1;
}
public static function findMatchIndices(array:Array,
element:Object, partialMatch:Boolean = false):Array {
        var indices:Array = new Array(    );
        var index:int = findMatchIndex(array,
                                       element,
                                       partialMatch);
        while(index != -1) {
            indices.push(index);
            index = findMatchIndex(array,
                                   element,
                                   partialMatch,
                                   index + 1);
        }
        return indices;
}
```

5.4. 删除数组元素

问题

我要删除一个或多个数组元素，或移动数组元素

解决办法

splice() 方法删除指定位置的元素，*pop()* 删除尾部元素，*shift()* 删除首部元素

讨论

删除指定位置的元素使用*splice()* 方法，它需要两个参数：

start

开始下标

deleteCount

删除的元素个数，如果没有定义，则从起始位置到末尾全部删除：

```
var letters:Array = ["a", "b", "c", "d"];
//从下标1开始删除1个元素
letters.splice(1, 1);
// 显示结果，现在只剩三个
// "a", "c", 和 "d".
for (var i:int = 0; i < letters.length; i++) {
    trace(letters [i]);
}
```


splice() 方法也返回一个新的包含删除的元素数组：

```
var letters:Array = ["a", "b", "c", "d"];
//删除两个元素，从0位置开始
var deleted:Array = letters.splice(0, 2);
// 显示: "a"， "b".
for (var i:int = 0; i < deleted.length; i++) {
    trace(deleted[i]);
}
```

删除首部和尾部的元素可用*shift()* 和*pop()* 方法。 *shift()* 方法删除首部第一个元素，然后返回该元素，*pop()* 方法删除尾部的元素并返回该值：

```
var letters:Array = ["a", "b", "c", "d"];
```

```
trace(letters.shift(    ));
trace(letters.pop(    ));
//显示剩下的元素
for (var i = 0; i < letters.length; i++) {
      trace(letters[i]);
}
```

在for循环里删除原始，也要修改下标值，下面的代码演示不更新下标值变量出现的情况：

```
var numbers:Array = new Array(4, 10);
numbers[4] = 1;
trace(numbers);    // 显示: 4,10,undefined,undefined,1
for(var i:int = 0; i < numbers.length; i++) {
      if(numbers[i] == undefined) {
            numbers.splice(i, 1);
      }
}
trace(numbers);    // 显示：  4,10,undefined,1
```

上面的代码本来是期望上出全部undefined 元素的，结果只删除一个，调试运行，看看发生了什么：

1。前两个循环什么都没做，因为都不是undefined.

2。第三次找到 undefined 然后删除它，这时，第4个和第5个元素下移变成了第3个和第4个元素

3。下一循环检测第4个元素，也就是最后一个，这时忽略了第3个元素也就是那个undefined 元素，因此，当删除元素，应该把小标变量-1，代码应该这样：

```
var numbers:Array = new Array(4, 10);
numbers[4] = 1;
trace(numbers);    // 显示: 4,10,undefined,undefined,1
for(var i:int = 0; i < numbers.length; i++) {
   if(numbers[i] == undefined) {
      numbers.splice(i, 1);
      i--;
   }
}
trace(numbers);    // 显示: 4,10,1
```

5.5. 在数组中间插入元素

问题

我要在数组中间插入元素u

解决办法

使用*splice()*方法

讨论

*splice()*方法不仅可以删除元素，也可以插入元素，插入的元素放到第2个参数之后，当第2个参数为0代表插入元素：

```
var letters:Array = ["a", "b", "c", "d"];
//插入三个元素，起始位置为1
letters.splice(1, 0, "r", "s", "t");
// letters 现在包含的元素有：
// "a", "r", "s", "t", "b", "c", "d".
for (var i:int = 0; i < letters.length; i++) {
    trace(letters[i]);
}
```

你也可以删除和插入同时执行：

```
var letters:Array = ["a", "b", "c", "d"];
//删除2个，插入3个
letters.splice(1, 2, "r", "s", "t");
// myArray 现在的元素
// "a", "r", "s", "t", and "d".
for (var i:int = 0; i < letters.length; i++) {
    trace(letters[i]);
}
```

5.6. 转换字符串为数组

问题

我有一堆字符串，想把它转换为数组。

解决办法

使用*String.split()* 方法

讨论

String 类的*split()* 方法把字符串转换为数组，但前提是字符串中含有统一的分割符，比如*Susan,Robert,Paula* 字符串分割符为逗号

split() 方法接受两个参数：

分割符

用分割符来分割字符串，如果没定义，则把整个字符串作为数组的第一个元素

数量

分割出的最大元素个数，如果没定义，则全部放入数组。

可以使用空格符作为分割符：

```
var list:String = "Peter Piper picked a peck of pickled peppers";
var words:Array = list.split(" ");
```

split() 方法在用*URLLoader* 对象读取数据时经常用到，比如你接受服务器的一些姓名字符串：

```
names=Michael,Peter,Linda,Gerome,Catherine
```

这是用*split()* 方法转换为数组：

```
// URLLoader 读取数据
var namesData:String = _loader.data;
var names:Array = namesData.split(",");
```

5.7. 转换数组为字符串

问题

我要把数组转换为字符串

解决办法

使用 *join()* 方法

讨论

ActionScript 提供内建的方法 *join()* 可以快速把数组转换为字符串（数组中的元素不管什么类型都将转换为字符串），该方法接受个参数作为分隔符：

```
var letters:Array = ["a", "b", "c"];
trace(letters.join("|"));     // 显示: a|b|c
```

如果不指定分隔符，默认为逗号：

```
var letters:Array = ["a", "b", "c"];
trace(letters.join());     // 显示: a,b,c
```

当 *join()* 的分隔符为逗号，其效果和*toString()* 一样。实际上当我们直接输出数组时系统就是调用*toString()* 方法进行转换的，例如：

```
var letters:Array = ["a", "b", "c"];
trace(letters);    // 显示: a,b,c
```

5.8. 创建数组的拷贝

问题

我要复制一份数组，内容完全一样，只是不同的引用

解决办法

使用 *concat()* 方法或 *slice()* 方法，另外还可以使用*ArrayUtilities.duplicate()* 方法，*duplicate()* 方法可以创建递归复制

讨论

因为数组是复合类型，因此它的比较和复制都和基本类型不同。一个变量指向数组但是实际上它并不包含数组数据，它只是指向内存中存放数组数据的位置。从优化的角度考虑，基本类型的占用空间往往很小，但是符合类型如数组可以变得很大，如果我们在日常操作中经常复制整个数组是非常不明智的，因此当你要复制数组时，ActionScript 并不是生成独立的一份拷贝，看下面的例子：

首先我们看看基本类型是怎么复制的：

```
// 赋值数字5给变量
var quantity:int = 5;
// 拷贝 quantity的值 给另一个变量 newQuantity.
var newQuantity:int = quantity;
// 改变 quantity的值
quantity = 29;
trace(quantity);            // 显示: 29
trace(newQuantity);         // 显示: 5
```

我们看到，两者是互不影响的，也就是说基本变量拷贝的是值

现在我们看看数组的操作，和上面的例子不同，两个变量实际上都指向了同一个数组存贮空间。当letters 变量改变数组内容时，也影响到newLetters变量：

```
// 赋值数组.
var letters:Array = ["a", "b", "c"];
// 拷贝 letters 到newLetters.
var newLetters:Array = letters;
// 两个数组包含相同内容
trace(letters);             // 显示: "a,b,c"
trace(newLetters);          // 显示: "a,b,c"
```

```
// 改变letters的值
letters = ["d", "e", "f"];
// 另一个数组也变了
trace(letters);              // 显示: "d,e,f"
trace(newLetters);           // 显示: "d,e,f" (而不是 "a,b,c")
```

其实这就像文件夹的两个快捷方式那样，两个快捷方式虽然名字不同，但是都指向同一个文件，无论哪个对文件夹操作，另一个快捷方式也发生变化。

如果真要复制数组，可以调用数组的 *concat()* 方法：

```
// 给数组赋值.
var letters:Array = ["a", "b", "c"];
//用concat( ) 创建新的数组
var newLetters:Array = letters.concat( );
// 两个数组内容一样
trace(letters);              // 显示: "a,b,c"
trace(newLetters);           // 显示: "a,b,c"
// 改变letters的值
letters = ["d", "e", "f"];
//不像上面的例子，这次两个数组内容不同了.
trace(letters);              // 显示: "d,e,f"
trace(newLetters);           // 显示: "a,b,c"
```

也可以用 *slice()* 方法代替 *concat()*， 例如：

```
var newLetters:Array = letters.slice(0);
```

concat() 或 *slice()* 方法复制一维整型下标的数组还可以，但是如果是多维的联合数组就不行了，对于联合数组,不能使用 *concat()* 或 *slice()* 方法，对于多维数组，用 *concat()* 或 *slice()* 只能复制顶层的数组，内部的就不能复制了，看下面的代码：

```
var coordinates:Array = new Array( );
coordinates.push([0,1,2,3]);
coordinates.push([4,5,6,7]);
coordinates.push([8,9,10,11]);
coordinates.push([12,13,14,15]);
// 复制
var coordinatesDuplicate:Array = coordinates.concat( );
```

```
// 替换元素
coordinatesDuplicate[0][0] = 20;
trace(coordinates[0][0]);   //显示: 20
// 替换顶层元素.
coordinatesDuplicate[1] = [21,22,23,24];
trace(coordinates[1]);   // 显示: 4,5,6,7
```

上面的代码coordinates 是个二维数组，coordinatesDuplicate 是coordinates的复制。但是，虽然是复制，有些元素仍然引用了原始数组元素，这就意味你改变了一个，会影响另一个数组的内容。实际上从上面的代码可以看出，只复制了顶层的数组，这一部分是互不关联的。

要想完全的复制数组，需要使用递归。*ArrayUtilities.duplicate()* 方法就是这样的做法，默认下它也只复制一层数组，当第2个参数设为true时则是递归复制：

```
// 创建二维数组
var coordinates:Array = new Array(    );
for(var i:int = 0; i < 4; i++) {
   coordinates[i] = new Array(    );
   for(var j:int = 0; j < 4; j++) {
      coordinates[i].push(String(i) + "," + String(j));
   }
}
// 复制coordinates.
var newCoordinates:Array = ArrayUtilities.duplicate(coordinates, true) as Array;
// 替换
newCoordinates[0][0] = "a";
// 使用toString() 方法输出
trace(ArrayUtilities.toString(coordinates));
trace(ArrayUtilities.toString(newCoordinates));
```

下面的例子演示用 *duplicate()* 方法复制联合数组：

```
var coordinatesMap:Object = new Object(    );
coordinatesMap.a = [{a: 1},{b: 2}, {c: 3}, {d: 4}];
coordinatesMap.b = [{a: 1},{b: 2}, {c: 3}, {d: 4}];
coordinatesMap.c = [{a: 1},{b: 2}, {c: 3}, {d: 4}];
coordinatesMap.d = [{a: 1},{b: 2}, {c: 3}, {d: 4}];
```

```
var newCoordinatesMap:Object = ArrayUtilities.duplicate(coordinatesMap, true);
newCoordinatesMap.a[0] = {r: 5};
trace(ArrayUtilities.toString(coordinatesMap));
trace(ArrayUtilities.toString(newCoordinatesMap));
```

两个例子可以看到，复制的数组改变不会影响原始数组。

5.9. 存储多维数据

问题

我要怎样存储多组相关的数据

解决办法

使用多维数组存储

讨论

除了一维数组，还可以创建多维数组，比如 *beginGradientFill()* 方法（在第7章讨论）使用三个平行数组表示colors, alphas, 和ratios ，每个数组都有相同的下标。

创建平行数组，然后操作同一下标的元素，使用平行数组，很容易访问同一下标的相关元素，比如：

```
var colors:Array = ["maroon", "beige",    "blue",       "gray"];
var years:Array  = [1997,       2000,      1985,          1983];
var makes:Array  = ["Honda",   "Chrysler", "Mercedes", "Fiat"];
// 循环这些数组，因此数组的长度相同，可以用任何一个数组的length属性，下面的例子使用了
//makes.length.
for (var i:int = 0; i < makes.length; i++) {
    trace("A " + colors[i] + " " +
                    years[i] + " " +
                    makes[i]);
    // 显示:
    // A maroon 1997 Honda
    // A beige 2000 Chrysler
    // A blue 1985 Mercedes
    // A gray 1983 Fiat
```

```
}
```

需要注意的是，如果改变了数组长度，必须同时修改其他数组。

另一种方法就是创建多维数组，它是数组的数组：

```
// 创建数组 cars，然后用数组组装填充，每个元素都是数组包含3个元素 (color, year, make).
var cars:Array = new Array();
cars.push(["maroon", 1997, "Honda"]);
cars.push(["beige", 2000, "Chrysler"]);
cars.push(["blue", 1985, "Mercedes"]);
cars.push(["gray", 1983, "Fiat"]);
// 循环遍历数组
for (var i:int = 0; i < cars.length; i++) {
    // 显示:
    // A maroon 1997 Honda
    // A beige 2000 Chrysler
    // A blue 1985 Mercedes
    // A gray 1983 Fiat
    TRace("A " + cars[i][0] + " " +
                    cars[i][1] + " " +
                    cars[i][2]);
}
```

下面的代码用二重循环遍历二维数组：

```
for (var i:int = 0; i < cars.length; i++) {
    for (var j:int = 0; j < cars[i].length; j++) {
        TRace("Element [" + i + "][" + j + "] contains: " +
                cars[i][j]);
    }
}
```

从上面的例子来看,很难区别cars[i][0] 和 cars[i][j]。但如果任何数组的长度发生变化这时cars[i][0]这种表示就很被动，需要修改代码才行。

另外对象数组的使用也很类似，只是多了个名称属性。对象数组用名称属性代替数字下标来索引元素：

```
// 创建数组cars，填充对象
```

```
// 每个对象有个make 属性，year 属性和 color 属性
var cars:Array = new Array();
cars.push({make: "Honda",        year: 1997, color: "maroon"});
cars.push({make: "Chrysler", year: 2000, color: "beige"});
cars.push({make: "Mercedes", year: 1985, color: "blue"});
cars.push({make: "Fiat",          year: 1983, color: "gray"});
// 遍历数组
for (var i:int = 0; i < cars.length; i++) {
        trace("A " + cars[i].color + " " +
                                cars[i].year + " " +
                                cars[i].make);
}
```

5.10. 数组排序

问题

我要进行数组排序

解决办法

使用 *sort()* 方法，对于对象数组可以用 *sortOn()* 方法

讨论

使用 *sort()* 方法就可以对数组进行排序，没有参数是进行升序排序，对于字符内容采用 Unicode 编码排序

```
var words:Array = ["tricycle", "relative", "aardvark", "jargon"];
words.sort(    );
trace(words); // 显示: aardvark,jargon,relative,tricycle
```

如果要进行降序排序，需要传递参数*Array.DESCENDING* 常量：

```
var words:Array = ["tricycle", "relative", "aardvark", "jargon"];
words.sort(Array.DESCENDING);
trace(words); // 显示: tricycle,relative,jargon,aardvark
```

上面的例子没有考虑大小写问题，比如：

```
var words:Array = ["Tricycle", "relative", "aardvark", "jargon"];
words.sort(    );
trace(words); // 显示: Tricycle,aardvark,jargon,relative
```

使用*Array.CASEINSENSITIVE* 常量忽略大小写进行排序：

```
var words:Array = ["Tricycle", "relative", "aardvark", "jargon"];
words.sort(Array.CASEINSENSITIVE);
trace(words); // 显示 aardvark,jargon,relative,Tricycle
```

如果对数字内容的数组排序，则根据第一个数字的ASCII 排序：

```
var scores:Array = [10, 2, 14, 5, 8, 20, 19, 6];
scores.sort(    );
trace(scores);     // 显示: 10,14,19,2,20,5,6,8
```

使用 *Array.NUMERIC* 常量才能正常对数字排序：

```
var scores:Array = [10, 2, 14, 5, 8, 20, 19, 6];
scores.sort(Array.NUMERIC);
```

```
trace(scores);     // 显示：2,5,6,8,10,14,19,20
```

还有两个常量*Array.UNIQUESORT* 和 a*rray.RETURNINDEXEDARRAY.* ，如果你只是对含有唯一元素的数组排序就可以用*Array.UNIQUESORT* ，Flash只会对这样的数组排序，不满足条件*sort()* 返回0，且不进行排序：

```
var ranking:Array = [2,5,6,3,1,1,4,8,7,10,9];
var sortedRanking:Object = ranking.sort(Array.UNIQUESORT);
trace(sortedRanking);     // 显示: 0
trace(ranking);    // 显示：2,5,6,3,1,1,4,8,7,10,9
```

Array.RETURNINDEXEDARRAY 得到排序后数组元素的下标顺序，但不改变原始数组：

```
var words:Array = ["tricycle", "relative", "aardvark", "jargon"];
var indices:Array = words.sort(Array.RETURNINDEXEDARRAY);
trace(words);     // 显示: tricycle,relative,aardvark,jargon
trace(indices); // 显示: 2,3,1,0
for(var i:int = 0; i < words.length; i++) {
   /* 显示
        aardvark
        jargon
        relative
        tricycle
   */
   trace(words[indices[i]]);
}
```


可以用操作符(|)联合使用这些常量：

```
var words:Array = ["Tricycle", "relative", "aardvark", "jargon"];
words.sort(Array.CASEINSENSITIVE | Array.DESCENDING);
trace(words);     // 显示: Tricycle,relative,jargon,aardvark
```

有时候你想反转数组该怎么办呢？*sort()* 方法并没这功能，这是可以用 *reverse()* 方法：

```
var words:Array = ["tricycle", "relative", "aardvark", "jargon"];
words.reverse(    );
trace(words);     // 显示: jargon,aardvark,relative,tricycle
```

上面的部分讨论了怎么对字符串和数字进行排序，对于对象数组可用 *sortOn()* 方法，该方法需要一个字符串参数指定名称属性对其排序：

```
var cars:Array = new Array();
cars.push({make: "Honda",       year: 1997, color: "maroon"});
cars.push({make: "Chrysler", year: 2000, color: "beige"});
cars.push({make: "Mercedes", year: 1985, color: "blue"});
cars.push({make: "Fiat",         year: 1983, color: "gray"});
// 对 year 属性进行排序 cars.sortOn("year")：
for (var i:int = 0; i < cars.length; i++) {
   /* 显示:
        gray       1983    Fiat
        blue       1985    Mercedes
        maroon    1997    Honda
        beige      2000    Chrysler
   */
   trace(cars[i].color + "\t" +
            cars[i].year + "\t" +
            cars[i].make);
}
```

sortOn() 方法也可以一次对多个字段进行排序，看下面的代码：

```
var cars:Array = new Array(    );
cars.push({make: "Honda",       year: 1997, color: "maroon"});
cars.push({make: "Chrysler", year: 2000, color: "beige"});
cars.push({make: "Mercedes", year: 1985, color: "blue"});
cars.push({make: "Fiat",         year: 1983, color: "gray"});
cars.push({make: "Honda",       year: 1992, color: "silver"});
cars.push({make: "Chrysler", year: 1968, color: "gold"});
cars.push({make: "Mercedes", year: 1975, color: "green"});
cars.push({make: "Fiat",         year: 1983, color: "black"});
cars.push({make: "Honda",       year: 2001, color: "blue"});
cars.push({make: "Chrysler", year: 2004, color: "orange"});
cars.push({make: "Mercedes", year: 2000, color: "white"});
cars.push({make: "Fiat",         year: 1975, color: "yellow"});
```

```
// 对两个字段排序
cars.sortOn(["year", "make"]);
for (var i:int = 0; i < cars.length; i++) {
   /* 显示:
        gold        1968       Chrysler
        yellow      1975       Fiat
        green       1975       Mercedes
        black       1983       Fiat
        gray        1983       Fiat
        blue        1985       Mercedes
        silver      1992       Honda
        maroon      1997       Honda
        beige       2000       Chrysler
        white       2000       Mercedes
        blue        2001       Honda
        orange      2004       Chrysler
   */
   trace(cars[i].color + "\t" +
            cars[i].year + "\t" +
            cars[i].make);
}
```

下面的例子，先对 make, 再对year排序：

```
cars.sortOn(["make", "year"]);
for (var i:int = 0; i < cars.length; i++) {
   /* 显示:
        gold        1968       Chrysler
        beige       2000       Chrysler
        orange      2004       Chrysler
        yellow      1975       Fiat
        black       1983       Fiat
        gray        1983       Fiat
```

```
        silver    1992       Honda
        maroon    1997       Honda
        blue      2001       Honda
        green     1975       Mercedes
        blue      1985       Mercedes
        white     2000       Mercedes
    */
    trace(cars[i].color + "\t" +
            cars[i].year + "\t" +
            cars[i].make);
}
```

sortOn() 方法也可用那些数组常量完成降序，忽略大小写等排序：

```
cars.sortOn("year", Array.DESCENDING);
for (var i:int = 0; i < cars.length; i++) {
    /* 显示:
        beige     2000    Chrysler
        maroon    1997    Honda
        blue      1985    Mercedes
        gray      1983    Fiat
    */
    trace(cars[i].color + "\t" +
            cars[i].year + "\t" +
            cars[i].make);
}
```

5.11. 实现自定义排序

问题

我要自定义数组排序

解决办法

把自定义比较的函数引用传递给*sort()* 方法

讨论

如果要自定义排序，可用*sort()* 方法和自定义比较函数。*sort()* 方法重复调用比较函数对两个数组元素进行比较，比较函数接受两个参数即数组元素（我们称为a和b），根据具体的排序方式返回正数，负数或0。如果返回负数，a排在b前，如果返回0，位置不变，如果返回正数，a排在b后，直到所有元素对比完毕。

下面有个例子对字符串数组进行自定义排序，比如是一个歌曲名数组，在排序时忽略字符串中含有的"The" 字母，首先看看默认的排序：

```
var bands:Array = ["The Clash",
                   "The Who",
                   "Led Zeppelin",
                   "The Beatles",
                   "Aerosmith",
                   "Cream"];
bands.sort(    );
for(var i:int = 0; i < bands.length; i++) {
    trace(bands[i]);
    /* 输出：
        Aerosmith
        Cream
        Led Zeppelin
        The Beatles
        The Clash
        The Who
    */
}
```

给 *sort()* 方法传递*bandNameSort* 比较函数：

```
var bands:Array = ["The Clash",
                   "The Who",
                   "Led Zeppelin",
                   "The Beatles",
                   "Aerosmith",
                   "Cream"];
bands.sort(bandNameSort);
for(var i:int = 0; i < bands.length; i++) {
    trace(bands[i]);
    /*输出
        Aerosmith
        The Beatles
        The Clash
        Cream
        Led Zeppelin
        The Who
    */
}
function bandNameSort(band1:String, band2:String):int
{
    band1 = band1.toLowerCase(    );
    band2 = band2.toLowerCase(    );
    if(band1.substr(0, 4) == "the ") {
        band1 = band1.substr(4);
    }
    if(band2.substr(0, 4) == "the ") {
        band2 = band2.substr(4);
    }
    if(band1 < band2) {
        return -1;
    }
```

```
    else {
        return 1;
    }
}
```

bandNameSort() 函数把字符串元素转换为小写，然后检测是否含有"The "，如果有则剪切掉，取剩余字符串进行比较

5.12. 数组元素的随机排序

问题

我要打乱数组元素的顺序

解决办法

使用 *sort()* 方法和自定义比较函数返回随机的正数或负数

讨论

很多情况我们需要得到一个随机排列的数组，比如有个游戏需要产生随机的字母。

有很多种方法达到这个目的，但是最简单的办法就是创建自定义比较函数，返回随机的正数或负数，把该函数引用传递给*sort()* 方法：

下面的比较函数就能达到目的：

```
function randomSort(elementA:Object, elementB:Object):Number {
    return Math.random(  ) - .5
}
```

Math.random( ) 返回0.0 到 1.0. 减去0.5 ，正好有一半的几率是负数，一半为正数，因此这个数组经过随机排序

看下面的随机排序例子：

```
var numbers:Array = new Array(  );
for(var i:int=0;i<20;i++) {
    numbers[i] = i;
}
numbers.sort(randomSort);
for(var i:int=0;i<numbers.length;i++) {
    trace(numbers[i]);
}
```

5.13. 取得数组元素的最大值和最小值

问题

我要获取数字数组的最大和最小元素

解决办法

经过数字排序，然后读取数组的第一个和最后一个元素

讨论

要想快速的取得最大值和最小值，先进行排序，看下面：

```
var scores:Array = [10, 4, 15, 8];
scores.sort(Array.NUMERIC);
trace("Minimum: " + scores[0]);
trace("Maximum: " + scores[scores.length - 1]);
```

如果不破坏原有数组顺序，可先复制数组：
也可使用 *ArrayUtilities.min()* 和 *ArrayUtilities.max()* 方法。

5.14. 比较数组

问题

我该怎么知道两个数组是否相等呢

解决办法

循环数组，一一比较对应位置的每个元素

讨论

因为数组是引用类型，使用=操作符只能对比引用是否指向同一内存空间，如：

```
var letters:Array = ["a", "b", "c", "d"];
var lettersPointer:Array = letters;
trace(letters == lettersPointer);   // 显示: true
```

但是如果数组内容相同，但是在不同的内存空间，=操作就会返回false：

```
var letters1:Array = ["a", "b", "c", "d"];
var letters2:Array = ["a", "b", "c", "d"];
trace(letters1 == letters2];   // 显示: false
```

因此，比较数组应该比较数组的每个元素是否相等：

```
var equivalent:Boolean = true;
for(var i:int = 0; i < letters1.length; i++) {
    if(letters1[i] != letters2[i]) {
        equivalent = false;
        break;
    }
}
trace(equivalent);   // 显示: true
```

另外还可以用*ArrayUtilities.equals()* 方法，该方法需要两个参数：两个数组引用，返回布尔值说明是否相等：

```
var letters1:Array = ["a", "b", "c", "d"];
var letters2:Array = ["a", "b", "c", "d"];
trace(ArrayUtilities.equals(letters1, letters2)); // 显示: true
```

默认，两个不同排列的数组是不相等的，除非提供第3个参数为true表示忽略数组排列顺序：

```
var letters1:Array = ["a", "b", "c", "d"];
```

```
var letters2:Array = ["b", "a", "d", "c"];
trace(ArrayUtilities.equals(letters1, letters2)); //  显示: false
trace(ArrayUtilities.equals(letters1, letters2, true)); //  显示: true
```

equals() 方法用起来很简单，下面是它的代码：

```
public static function equals(arrayA:Array,    arrayB:Array,
                                        bNotOrdered:Boolean):Boolean {
    // 如果两个数组长度不同
    if(arrayA.length != arrayB.length) {
        return false;
    }
    // 创建拷贝，不影响原数组
    var arrayACopy:Array = arrayA.concat(    );
    var arrayBCopy:Array = arrayB.concat(    );
    // 如果忽略排列顺序
    if(bNotOrdered) {
        arrayACopy.sort(    );
        arrayBCopy.sort(    );
    }
    // 循环比较，如果不匹配，删除拷贝，返回false
    for(var i:int = 0; i < arrayACopy.length; i++) {
        if(arrayACopy[i] != arrayBCopy[i]) {
            delete arrayACopy;
            delete arrayBCopy;
            return false;
        }
    }
    // 否则相等，删除数组，返回true
    delete arrayACopy;
    delete arrayBCopy;
    return true;
}
```

5.15. 创建关联数组

问题

我要创建用名称元素作为索引的数组

解决办法

创建关联数组

讨论

用关联数组其每个元素都有特定的含义，这一点原来的数组类型是做不到的。

```
var aMembers:Array = new Array("Franklin", "Gina", "Sindhu");
```

关联数组在其他的语言叫做哈希表，在 ActionScript 里它就是*Object* 类的一个实例，关联数组使用名称元素来代替数字下标，该名称也被称为关键字或属性，说关键字更好理解些，它关联了元素值，两者一一对应。

创建关联数组不是用*Array* 类而是Object类创建的，它就是 *Object* 类的一个实例，理论上*Object* 类是任何类的基类。所有的对象都能作为关联数组，但是除非有特殊需要，最好还是用*Object* 类创建。

用{ } ，而且用逗号分开每个键值对，键值对之间用:，像下面：

```
var memebers:Object = {scribe: "Franklin",
                       chairperson: "Gina",
                       treasurer: "Sindhu"};
```

也可以像下面那样创建关联数组：

```
var members:Object = new Object( );
members.scribe = "Franklin";
members.chairperson = "Gina";
members.treasurer = "Sindhu";
```

有两种方法访问关联数组内容，一种是通过访问属性名称（关键字）：

```
trace(members.scribe); // 显示: Franklin
```

另一种就像数组那样，把关键字作为下标来访问，用[ ] 符号：

```
trace(members["scribe"]); // 显示: Franklin
```

这种方式更加灵活，可以在数组中进行遍历，对于动态生成的关键值和内容这种访问方式是最好的，例如：

```
var members:Object = new Object();
members.councilperson1 = "Beatrice";
```

```
members.councilperson2 = "Danny";
members.councilperson3 = "Vladamir";
for (var i:int = 1; i <= 3; i++) {
        trace(members["councilperson" + i];
}
```

数组访问方式在循环语句里经常用到：

```
var members:Object = new Object(    );
members["councilperson"] = "Ruthie";
trace(members.councilperson);                // 显示  Ruthie
members.councilperson = "Rebecca";
trace(members["councilperson"]);          // 显示: Rebecca
```

5.16. 读取关联数组

问题

我要怎样遍历关联数组

解决办法

使用 *for . . . in* 语句

讨论

基于整形下标的数组可以通过*for* 语句进行循环遍历，但是，用关键字作索引的关联数组就不能这样遍历了，还好，关联数组可以通过 *for . . . in* 语句进行遍历访问。该语句会访问指定对象所有可用的属性，语法如下：

```
for (key in object) {
    // Actions
}
```

for . . . in 语句不需要循环变量更新语句，决定循环次数的是对象的属性个数。注意这*key* 就是存储每个属性名称的：

```
var members:Object = new Object(    );
members.scribe = "Franklin";
members.chairperson = "Gina";
members.treasurer = "Sindhu";
// 使用 for . . . in 语句遍历所有元素
for (var sRole:String in members) {
    // 显示:
    // treasurer: Sindhu
    // chairperson: Gina
    // scribe: Franklin
    trace(sRole + ": " + members[sRole]);
}
```

6.0. 简介

ActionScript 3.0 和 Flash Player 9 的渲染模型已经和以前的版本有很大不同。以前 *MovieClip* 是渲染的焦点。*swf* 电影的根节点就是 *MovieClip* (现在用*Stage*). 根节点*MovieClip* 可以包含子节点 *MovieClip*s， 子节点还可包含更多的子节点*MovieClip*s。这样的层次结构用来控制 *MovieClip*s 的绘制（深度越深表示显示在最顶层）。*createEmptyMovieClip(), attachMovie(),* 或 *duplicateMovie- Clip()* 用来创建*MovieClip*s。一旦*MovieClip* 被创建，它就自动添加到这个可视化层级列表中通过渲染器进行绘制。*MovieClip*s 在层级列表中不能移动位置，非要这么做则只有删除当前的，然后在指定位置创建。

新的渲染器仍基于层级结构，但是于之前相比是经过最优化处理，更简单和灵活。新的渲染模型类都集中在 *flash.display* 包中。这个包中包含所有在*.swf* 电影中用到的可视化类。其他不在该包中的任何对象都不会被渲染器绘制。每个*.swf* 电影包含一个可视化对象列表，有下面三种类型：

stage

stage 是可视化对象列表层级的根节点。每个电影有一个stage 对象，它包含屏幕上显示出的所有对象。实际上stage 是一个容器，可以通过引用stage属性来访问任何可视化对象。

可视化对象容器

可视化对象容器包含其他可视化对象，stage 就是个可视化对象容器。其他的可视化对象容器包括 *Sprite, MovieClip,* 和 *Shape*。当一个可视化对象容器被删除时，它包含的所有子对象都将被删除。

可视化对象

可视化对象就是一个可显示的元素。有两种一种是可视化对象容器，比如*MovieClip*，另外是可视化对象，比如一个*TextField*。可视化对象创建后不会立即被显示出来，只有被添加到可视化对象容器后才被显示。

可视化对象列表进行了以下方面的改进：

提高了性能

可视化对象列表除了*MovieClip*还包含了多个可视化类。如*Sprite* 可以减少内存需求，另外*Shape* 可以代替*MovieClip* 实例进行绘画。用这些轻量级的类节省了内存和资源，提高了性能。

更容易的层级管理

新的可视化对象列表模型增加了层级管理功能。在以前的版本中，通过调用 *getNextHighestDepth()* 在当前位置创建*MovieClip*s，*swapDepths()* 控制层级顺序。这种层级管理是非常不合理和不灵活，写程序时也要非常小心。新的可视化对象列表模型直接自动处理了这个问题，彻底摆脱了手动层级管理。

灵活的结构

原来的模型结构非常死板和不灵活，要修改层级， *MovieClip*s 必须删除然后再在新位置创建，这简直是在浪费时间，也直接导致了运行速度下降。新的显示模型具备高灵活性，可以随意移

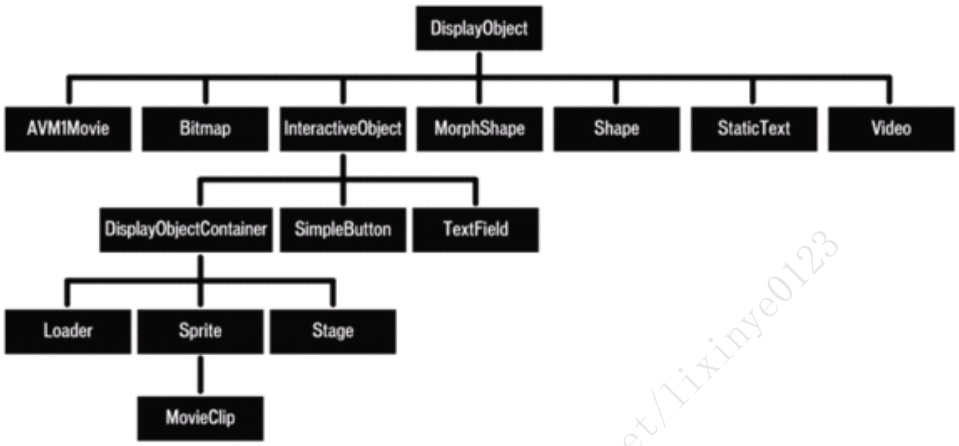
动元素到任何位置而不需要删除再创建。

创建可视化对象更容易

新的渲染模型更容易创建可视化对象，特别是自定义可视化类的创建。之前的模型需要扩展*MovieClip*，再连接指定的类库。*attachMovie()* 只能用来创建自定义类实例。新的模型，可以扩展更多的可视化类，这些将在第6.4节讨论。

flash.display 包包含了核心的可视化类，老的模型的核心类是*MovieClip*，现在*DisplayObject* 类顶级可视化类以及它的各种子类。

类层级图：



每个核心类都有特定功能，这样的层级类提供了更加灵活的编程能力（相对于以前只有*MovieClip*）。

6.1. 在可视化对象列表中添加项目

问题

我要添加新的可视化对象在屏幕上显示

解决办法

使用*DisplayObectContainer*类的*addChild()* 和*addChildAt()* 方法

讨论

Flash Player 现在把ActionScript 虚拟机(AVM)和渲染引擎两块功能集成在一起了。AVM 完成执行ActionScript 代码，而渲染引擎负责在屏幕上绘制对象，绘制对象需要两步骤：

通过ActionScript引擎创建可视化对象。
渲染引擎把可视化对象绘制在屏幕上。

第一步用new操作符创建可视化对象实例，任何直接或间接继承自*DisplayObject*类的实例都可以添加到可视化对象列表中，比如*Sprite*，*MovieClip*，*TextField*，或自定义类。用下列代码创建*TextField*：

```
var hello:TextField = new TextField(    );
```

上面的代码在AVM中创建了一个*TextField* display 对象，但是该对象没有绘制在屏幕上，因为它还不存在于渲染引擎里，要把它放到渲染引擎里需要添加该对象到可视化对象列表，通过调用*DisplayObjectContainer*实例的*addChild()*或*addChildAt()* 方法添加。

addChild() 方法接受一个参数，就是要显示的对象，下面的代码演示如何在AVM中创建对象，然后通过渲染引擎显示它：

```
package {
   import flash.display.DisplayObjectContainer;
   import flash.display.Sprite;
   import flash.text.TextField;
   public class DisplayListExample extends Sprite {
      public function DisplayListExample(    ) {
         // 创建对象
         var hello:TextField = new TextField(    );
         hello.text = "hello";
         //显示对象
         addChild( hello );
      }
```

```
    }
}
```

这里*DisplayListExample* 类是.swf电影的主类，它继承自*Sprite* 类。因此也间接继承自*DisplayObjectContainer*，它允许使用*addChild()* 方法添加对象作为其子对象。

TextField 对象通过*DisplayListExample* 构造函数创建，它是在AVM中创建，这时该对象不会立即显示在屏幕上因为渲染引擎还不知道它呢，只有通过*addChild()*添加它到可视化对象列表中*TextField* 才被显示。

*addChild()*和*addChildAt()* 方法把添加的对象作为可视化对象容器的子对象。

下面的代码演示*addChild()* 方法把对象添加到容器上而不是添加到可视化列表，因为容器对象不在可视化列表里，所以这个文本是不可见的：

```
// 创建文本框显示文本
var hello:TextField = new TextField(    );
hello.text = "hello";
// 创建容器
var container:Sprite = new Sprite(    );
// 添加TextField 作为容器的子对象
container.addChild( hello );
```

要显示文本，需要把container 添加到可视化对象列表中。通过列表的引用如root或stage，调用其*addChild()*，container 作为该方法的参数，root 和stage 都是*DisplayObject* 类的属性：

```
DisplayObjectContainer( root ).addChild( container );
```

可视化对象容器能容纳多个子对象，并且管理着对象列表的排列顺序，这个顺序决定这些对象显示在屏幕上的顺序。每个子对象在列表中都有个位置，它用一个整数来索引，这很像数组，位置0代表在列表的底层，它在位置1下绘制，位置1在位置2下绘制，依次类推，如果以前编写过Flash程序，这个深度概念是很容易理解的。

当通过*addChild()* 方法添加的对象会显示在最顶层，也就是它得到最大的位置索引值，如果要自己指定位置可使用*addChildAt()* 方法。

*addChildAt()*方法需要两个参数：要显示的对象，和队列位置。如果指定为0则会显示在最底下，如果指定为某个对象的后面，则它前面的对象都要向前移动，如果指定非法值比如负数或很大的数，就会抛出*RangeError* 错误。

下面的例子创建三个不同颜色的圆，红色和蓝色的圆通过*addChild()* 方法添加，蓝色的在前，因为它比红色迟些加入的，现在红色圆的位置为0，蓝色的为1。绿色的圆要通过*addChildAt()* 方法查到红色和蓝色之间，指定位置为1，这时蓝色的位置前移变成2。最后的顺序为红色为0，绿色为1，蓝色为2：

```
package {
    import flash.display.*;
```

```
    public class CircleExample extends Sprite {
        public function CircleExample(    ) {
            var red:Shape = createCircle( 0xFF0000, 10 );
            red.x = 10;
            red.y = 20;
            var green:Shape = createCircle( 0x00FF00, 10 );
            green.x = 15;
            green.y = 25;
            var blue:Shape = createCircle( 0x0000FF, 10 );
            blue.x = 20;
            blue.y = 20;
            addChild( red );
            addChild( blue );
            addChildAt( green, 1 );
        }
        public function createCircle( color:uint, radius:Number ):Shape {
            var shape:Shape = new Shape(    );
            shape.graphics.beginFill( color );
            shape.graphics.drawCircle( 0, 0, radius );
            shape.graphics.endFill(    );
            return shape;
        }
    }
}
```

现在我们只讨论了如何添加，那么当重复添加同一个对象会发生什么事呢？这时，它会先删除原来已经添加的对象，然后再把对象添加进去。

下面的例子演示：一个容器被创建来显示红色，绿色和蓝色圆，然后容器被添加到显示列表中。另一个容器创建添加到显示列表中，把红色圆移到第2个容器上，因此第2个容器显示在第1个上面，所以红色圆显示在最上面。

```
package {
    import flash.display.*;
    public class DisplayListExample extends Sprite {
```

```
        public function DisplayListExample(    ) {
            var red:Shape = createCircle( 0xFF0000, 10 );
            red.x = 10;
            red.y = 20;
            var green:Shape = createCircle( 0x00FF00, 10 );
            green.x = 15;
            green.y = 25;
            var blue:Shape = createCircle( 0x0000FF, 10 );
            blue.x = 20;
            blue.y = 20;
            var container1:Sprite = new Sprite(    );
            container1.addChild( red );
            container1.addChild( green );
            container1.addChild( blue );
            addChild( container1 );
            var container2:Sprite = new Sprite(    );
            addChild( container2 );
            container2.addChild( red );
        }
        public function createCircle( color:uint, radius:Number ):Shape {
            var shape:Shape = new Shape(    );
            shape.graphics.beginFill( color );
            shape.graphics.drawCircle( 0, 0, radius );
            shape.graphics.endFill(    );
            return shape;
        }
    }
}
```

6.2. 从显示列表中删除项目

问题

我要从可视化对象列表删除可视化对象，从屏幕上消失。

解决办法

使用*DisplayObectContainer*类的*removeChild()*和*removeChildAt()*方法

讨论

6.1节 讨论了如何使用*addChild()*和*addChildAt()* 方法添加可视化对象到可视化对象列表中。与此相反*removeChild()*和*removeChildAt()* 方法删除对象。

removeChild() 方法需要一个参数，那就是将要删除的对象引用。如果该对象不是可视化容器中的对象，就会抛出 *ArgumentError* 异常：

```
package {
   import flash.display.Sprite;
   import flash.text.TextField;
   import flash.events.MouseEvent;
   public class RemoveChildExample extends Sprite {
      private var _label:TextField;
      public function RemoveChildExample(    ) {
         _label = new TextField(    );
         _label.text = "Some Text";
         addChild( _label );
         stage.addEventListener( MouseEvent.CLICK, removeLabel );
      }
      public function removeLabel( event:MouseEvent ):void {
         removeChild( _label );
      }
   }
}
```


上面的代码创建个本地变量保存*TextField*的引用，这很必要，因为*removeChild()* 方法需要这个参数，如果没有这个变量，*removeChild()*方法就不可用了，这时可以用*removeChildAt()* 方法。

removeChildAt() 方法就像*addChildAt()* 方法一样需要一个数字参数，就是对象在队列中的位置，该值的合法范围在0到列表中的子对象数目。如果该值不合法则会抛出*RangeError*异常，下

面的例子使用了*removeChildAt()*方法

```
package {
   import flash.display.Sprite;
   import flash.text.TextField;
   import flash.events.MouseEvent;
   public class DisplayListExample extends Sprite {
      public function DisplayListExample(    ) {
         var label:TextField = new TextField(    );
         label.text = "Some Text";
         addChild( label );
         stage.addEventListener( MouseEvent.CLICK, removeLabel );
      }
      public function removeLabel( event:MouseEvent ):void {
         if ( numChildren > 0 ) {
            removeChildAt( 0 );
         }
      }
   }
}
```

使用*removeChildAt( )*最大的不同就是不再需要先申明*TextField*的引用变量了，在*RemoveChildExample*容器中，它被添加到位置0处，要删除它只要指定位置为0，传递0给*removeChildAt()*。

注意，从列表中移出并不算完全删除了，要完全删除需要把所有指向它的引用设为null。

如果要移除所有容器中的子对象，把*removeChildAt( )*放在*for*循环中，每个容器都有个numChildren 属性表示它所拥有的子对象数目。

当一个子对象被移除，比它高的子对象将会全部下移一位。比如说一个容器有3个子对象他们在0, 1,和2的位置。当在位置0的子对象被移除，其他对象位置自动减1。这里有两种方式移除对象：

总是移除位置0的对象。
移除最上面的对象。

第一种情况，每次移除第一个位置的对象，其他对象下移，接着还是移除第一个位置的对象，循环反复。

第二种情况，移除最后的对象不会导致其他对象重新调整位置。

ascb.util.DisplayObjectUtilities 类实现了第一种方式：

```
package ascb.util {
   import flash.display.*;
   public class DisplayObjectUtilities {
      // 删除所有对象
      public static function removeAllChildren(
                                          container:DisplayObjectContainer ):void {
         var count:int = container.numChildren;
         for ( var i:int = 0; i < count; i++ ) {
            container.removeChildAt( 0 );
         }
      }
   }
}
```

使用*DisplayObjectUtilities.removeAllChildren()* 方法更直接些，看下面：

```
package {
   import flash.display.*;
   import ascb.util.DisplayObjectUtilities;
   public class DisplayListExample extends Sprite {
      public function DisplayListExample(    ) {
         addChild( new Sprite(    ) );
         addChild( new Sprite(    ) );
         // 删除所有子对象
         DisplayObjectUtilities.removeAllChildren( this );
         // 演示所以子对象都被删除了，显示0
         trace( numChildren );
      }

   }
}
```

6.3. 向前或向后移动对象

问题

我要改变对象在屏幕上的显示顺序

解决办法

使用*DisplayObectContainer*类的*setChildIndex( )* 方法改变项目的位置，*getChildIndex( )* 和*getChildAt( )* 方法得到项目在显示列表中的位置

讨论

6.1 节和 6.2 节 介绍了可视化对象列表如何处理堆栈顺序（深度），基本上每个*DisplayObjectContainer* 实例都有一个子对象列表，列表中子对象的顺序代表在屏幕上的绘制顺序，每个子对象有个位置索引，范围从0到numChildren -1，这很像数组。位置0的对象绘制在最底层，该列表不会出现空位置，比如三个对象分别在 0，1，2 (不会出现0，1，6).

*DisplayObjectContainer*的*setChildIndex( )* 方法用来重新设置列表中对象的位置，它需要两个参数：对象引用和合法的位置。非法的位置会抛出*RangeError* 异常。

下面的例子创建了三个不同颜色的圆，蓝色最上面，现在用*setChildIndex( )* 方法把蓝色的圆移动最底下去：

```
package {
   import flash.display.*;
   public class SetChildIndexExample extends Sprite {
      public function SetChildIndexExample(    ) {
         // 创建三个圆
         var red:Shape = createCircle( 0xFF0000, 10 );
         red.x = 10;
         red.y = 20;
         var green:Shape = createCircle( 0x00FF00, 10 );
         green.x = 15;
         green.y = 25;
         var blue:Shape = createCircle( 0x0000FF, 10 );
         blue.x = 20;
         blue.y = 20;
         addChild( red );
         addChild( green );
```

```
            addChild( blue );
            // 把蓝色圆移到最底层
            setChildIndex( blue, 0 );
        }
        public function createCircle( color:uint, radius:Number ):Shape {
            var shape:Shape = new Shape(    );
            shape.graphics.beginFill( color );
            shape.graphics.drawCircle( 0, 0, radius );
            shape.graphics.endFill(    );
            return shape;
        }
    }
}
```

setChildIndex() 方法需要明确知道把子对象移到哪里，把它移到最底层就指定为0，把它移动到最顶层就指定为numChildren-1，但如果我要移动到某个对象之后呢？

举个例子，假设我有两个圆，一个绿色一个蓝色，事先我不知道他们的位置，现在我要把蓝色的移动到绿色的前面，*setChildIndex()* 就做不到了，解决的办法就是用*getChildIndex()*方法 获得对象的位置，然后用*setChildIndex()*方法设置，*getChildIndex()* 方法只能返回队列中对象位置，否则就会抛出*ArgumentError* 异常

下面的例子创建了两个圆，绿色和蓝色的，*getChildIndex()*得到绿色圆的位置来设置蓝色圆的位置：

```
package {
    import flash.display.*;
    public class GetChildIndexExample extends Sprite {
        public function GetChildIndexExample(    ) {
            // 创建圆
            var green:Shape = createCircle( 0x00FF00, 10 );
            green.x = 25;
            green.y = 25;
            var blue:Shape = createCircle( 0x0000FF, 20 );
            blue.x = 25;
            blue.y = 25;
            addChild( green );
```

```
            addChild( blue );
            // 把蓝色圆放在绿色圆的后面
            setChildIndex( blue, getChildIndex( green ) );
        }
        public function createCircle( color:uint, radius:Number ):Shape {
            var shape:Shape = new Shape(    );
            shape.graphics.beginFill( color );
            shape.graphics.drawCircle( 0, 0, radius );
            shape.graphics.endFill(    );
            return shape;
        }
    }
}
```

如果对象a在b前，下面的代码直接把a移到b后面：

```
setChildIndex( a, getChildIndex( b ) );
```

如果a在b后面，上面的代码就把a移到b前面了。

如果事先没获得子对象引用的话就只能用*getChildAt()* 方法了。

getChildAt() 方法需要一个参数既列表子对象位置，返回该对象引用，如果给出的位置不合法则抛出*RangeError* 异常。

下面的例子创建了多个各种颜色，不同大小不同位置的圆，每次鼠标点击最底层的子对象就会移到最上层：

```
package {
    import flash.display.*;
    import flash.events.*;
    public class GetChildAtExample extends Sprite {
        public function GetChildAtExample(    ) {
            var color:Array = [ 0xFF0000, 0x990000, 0x660000, 0x00FF00,
                                0x009900, 0x006600, 0x0000FF, 0x000099,
                                0x000066, 0xCCCCCC ];
            for ( var i:int = 0; i < 10; i++ ) {
                var circle:Shape = createCircle( color[i], 10 );
                circle.x = i;
```

```
            circle.y = i + 10; // the + 10 adds padding from the top
            addChild( circle );
        }
        stage.addEventListener( MouseEvent.CLICK, updateDisplay );
    }
    public function updateDisplay( event:MouseEvent ):void {
        setChildIndex( getChildAt(0), numChildren - 1 );
    }
    public function createCircle( color:uint, radius:Number ):Shape {
        var shape:Shape = new Shape(    );
        shape.graphics.beginFill( color );
        shape.graphics.drawCircle( 0, 0, radius );
        shape.graphics.endFill(    );
        return shape;
    }
  }
}
```

6.4. 创建自定义可视化类

问题

我要创建新的可视化类

解决办法

继承*DisplayObject* 或它的子类来创建新类

讨论

基于新的可视化对象列表模型创建新的类是非常方便的。这在过去，只有继承*MovieClip* 才能创建新的类，现在完全不同了，新的可视化模型处理起来更简单，用ActionScript代码可以做更多的事情了。

新的可视化模型，正如介绍里讨论过的那样，现在除了*MovieClip*.还有很多可视化类可用。在你创建新类前，你需要决定的是创建什么类型的类。比如说你要创建一个图形类，这就要继承*Shape*类，如果你要创建自定义按钮，你可能要继承*SimpleButton*类。如果你要创建一个容纳其他对象的容器，*Sprite* 就是个很好的继承类，它不需要使用时间线，如果你需要时间线，需要继承*MovieClip*。

所有的可视化类都有其特定用途，当你创建类时首先要决定新的类是什么用途，然后选择相应的基类继承之。选择什么做父类也是很讲究的，不然会造成性能下降和资源浪费。比如一个简单的*Circle* 类不需要继承自*MovieClip* ，应该不需要时间线，*Shape* 类才是作为父类的最后选择。

基类写好了，接着就该写代码了，看下面的例子：创建了一个*Circle* 类，继承自*Shape* 类，文件名为*Circle.as*：

```
package {
   import flash.display.Shape;
   public class Circle extends Shape {
      private var _color:uint;
      private var _radius:Number;
      public function Circle( color:uint = 0x000000, radius:Number = 10 ) {
         _color = color;
         _radius = radius;
         draw(    );
      }
      private function draw(    ):void {
         graphics.beginFill( _color );
         graphics.drawCircle( 0, 0, _radius );
```

```
            graphics.endFill(    );
        }
    }
}
```

上面的代码定义了新的*Circle* 可视化类，创建*Circle* 实例时可以指定颜色和半径。
现在只要把*Circle* 类实例用*addChild()* 或 *addChildAt()* 加到可视化对象列表中就可以在屏幕上显示了。下面的代码例子创建了*Circle* 实例并显示在屏幕上：

```
package {
    import flash.display.Sprite;
    public class UsingCircleExample extends Sprite {
        public function UsingCircleExample(    ) {
            var red:Circle = new Circle( 0xFF0000, 10 );
            red.x = 10;
            red.y = 20;
            var green:Circle = new Circle( 0x00FF00, 10 );
            green.x = 15;
            green.y = 25;
            var blue:Circle = new Circle( 0x0000FF, 10 );
            blue.x = 20;
            blue.y = 20;
            // Add the circles to the display list
            addChild( red );
            addChild( green );
            addChild( blue );
        }
    }
}
```

6.5. 创建简单的按钮

问题

我想创建一个交互式按钮，点击鼠标执行一段代码，比如发送表单或计算结果

解决办法

创建*SimpleButton* 类实例和创建upState, downState, overState, 和 hitTestState等对象。当用户点击按钮时用click事件激活方法

讨论

可视化对象列表模型提供一种简单的方法通过*SimpleButton* 类创建按钮。*SimpleButton* 类允许用户用鼠标和可视化对象进行交互，通过各种状态定义交互方法，*SimpleButton* 按钮的状态有以下这些：

upState

代表按钮默认"up"状态的对象，当鼠标不在按钮上，按钮就处于这个状态。

overState

当鼠标移动到按钮上时按钮的状态，鼠标离开时按钮又回到"up"状态。

downState

当鼠标按下左键时所处的状态

hitTestState

这个状态定义按钮的界限，只是用来跟踪鼠标的目的。

下面的例子创建了一个*SimpleButton* 实例，定义了四个状态属性。每个状态都需要一个*DisplayObject* 实例相关联，*createCircle()* 方法创建不同颜色的图形作为按钮状态：

```
package {
   import flash.display.*;
   import flash.events.*;
   public class SimpleButtonDemo extends Sprite {
      public function SimpleButtonDemo(    ) {
         var button:SimpleButton = new SimpleButton(    );
         button.x = 20;
         button.y = 20;
         button.upState = createCircle( 0x00FF00, 15 );
         button.overState = createCircle( 0xFFFFFF, 16 );
         button.downState = createCircle( 0xCCCCCC, 15 );
```

```
            button.hitTestState = button.upState;
            button.addEventListener( MouseEvent.CLICK, handleClick );
            addChild( button );
        }
        private function createCircle( color:uint, radius:Number ):Shape {
            var circle:Shape = new Shape(    );
            circle.graphics.lineStyle( 1, 0x000000 );
            circle.graphics.beginFill( color );
            circle.graphics.drawCircle( 0, 0, radius );
            circle.graphics.endFill(    );
            return circle;
        }
        private function handleClick( event:MouseEvent ):void {
            trace( "Mouse clicked on the button" );
        }
    }
}
```

运行上面的代码后，绿色圆显示在屏幕上。当你的鼠标移动到绿色圆上时，白色圆现实出来了，点击这个白色圆，白色圆变成了灰色圆，这就是*SimpleButton* 实例通过不同的状态显示不同的对象。

addEventListener() 方法注册鼠标事件和处理函数，上面的代码注册了MouseEvent.CLICK事件，也就是鼠标单击事件，激活该事件后由*handleClick()* 方法处理，任何时候只要用户点击该按钮，*handleClick()* 方法就被激活，在这里编写处理代码，这里只是简单的输出个字符串显示在控制台上。

hitTestState 属性可能最有意思了，我们注意到上面的代码中把hitTestState 和upState设成了一样的状态。也就说当鼠标进入upState 对象范围大小时就激活事件。

hitTestState 可以被设置成任何可显示的对象，像下面的代码那样把激活区域设大些：

```
button.hitTestState = createCircle( 0x000000, 50 );
```

再次运行你会发现，当鼠标还没靠近圆，over状态就激活了，这就是激活半径现在是50的缘故。

下面继承*SimpleButton* 创建可视化类，包括定义四个状态，命名为*RectangleButton*：

```
package {
    import flash.display.*
    import flash.text.*;
```

```
import flash.filters.DropShadowFilter;
public class RectangleButton extends SimpleButton {
    // 显示在按钮上的文本
    private var _text:String;
    // 保存矩形的宽度和高度
    private var _width:Number;
    private var _height:Number;
    public function RectangleButton( text:String, width:Number, height:Number ) {
        // Save the values to use them to create the button states
        _text = text;
        _width = width;
        _height = height;
        // 创建按钮状态
        upState = createUpState(    );
        overState = createOverState(    );
        downState = createDownState(    );
        hitTestState = upState;
    }
    // 创建状态对象
    private function createUpState(    ):Sprite {
        var sprite:Sprite = new Sprite(    );
        var background:Shape = createdColoredRectangle( 0x33FF66 );
        var textField:TextField = createTextField( false );
        sprite.addChild( background );
        sprite.addChild( textField );
        return sprite;
    }
    private function createOverState(    ):Sprite {
        var sprite:Sprite = new Sprite(    );
        var background:Shape = createdColoredRectangle( 0x70FF94 );
        var textField:TextField = createTextField( false );
```

```
            sprite.addChild( background );
            sprite.addChild( textField );
            return sprite;
        }
        private function createDownState(    ):Sprite {
            var sprite:Sprite = new Sprite(    );
            var background:Shape = createdColoredRectangle( 0xCCCCCC );
            var textField:TextField = createTextField( true );
            sprite.addChild( background );
            sprite.addChild( textField );
            return sprite;
        }
        private function createdColoredRectangle( color:uint ):Shape {
            var rect:Shape = new Shape(    );
            rect.graphics.lineStyle( 1, 0x000000 );
            rect.graphics.beginFill( color );
            rect.graphics.drawRoundRect( 0, 0, _width, _height, 15 );
            rect.graphics.endFill(    );
            rect.filters = [ new DropShadowFilter( 2 ) ];
            return rect;
        }
        // 创建按钮上的文字
        private function createTextField( downState:Boolean ):TextField {
            var textField:TextField = new TextField(    );
            textField.text = _text;
            textField.width = _width;
            var format:TextFormat = new TextFormat(    );
            format.align = TextFormatAlign.CENTER;
            textField.setTextFormat( format );
//垂直居中
            textField.y = ( _height - textField.textHeight ) / 2;
```

```
            textField.y -= 2;    // Subtract 2 pixels to adjust for offset
            if ( downState ) {
                textField.x += 1;
                textField.y += 1;
            }
        }
    }
}
```

现在已经完成了*RectangleButton* 类的封装设计，只需要调用构造函数就可以创建实例了，看下面的代码演示：

```
package {
    import flash.display.*;
    public class SimpleButtonDemo extends Sprite {
        public function SimpleButtonDemo(    ) {
            // 创建三个不同文字不同大小和位置的矩形按钮
            var button1:RectangleButton = new RectangleButton( "Button 1", 60, 100 );
            button1.x = 20;
            button1.y = 20;
            var button2:RectangleButton = new RectangleButton( "Button 2", 80, 30 );
            button2.x = 90;
            button2.y = 20;
            var button3:RectangleButton = new RectangleButton( "Button 3", 100, 40 );
            button3.x = 100;
            button3.y = 60;
            addChild( button1 );
            addChild( button2 );
            addChild( button3 );
        }
    }
}
```

6.6. 动态载入外部图片

问题

我要在Flash运行时动态载入图片

解决办法

使用新的*Loader*类载入图片（jpg，png，gif）

讨论

9.17节 将展示如何在编译期通过[Embed] 元数据标签绑定外部文件到Flash。在运行期间载入外部图片或flash需要用到*Loader* 类。

flash.display.Loader 类非常类似于*flash.net.URLLoader* 类（19.3节讨论）。不同的是*Loader* 实例能载入外部图片和flash，在传输数据方面*URLLoader*更有用些。

载入外部内容需要三个步骤：

创建*Loader* 类实例
把*Loader* 实例加到显示列表里
调用*load()*方法载入外部内容

load() 方法下载图片或.swf文件，它需要一个*URLRequest* 对象作为参数，该对象指定一个需要下载资源的URL。

下面的例子通过*Loader* 实例下载一张图片*image.jpg* ：

```
package {
   import flash.display.*;
   import flash.net.URLRequest;
   public class LoaderExample extends Sprite {
      public function LoaderExample(    ) {
         // 1. 创建Loader 类实例
         var loader:Loader = new Loader(    );
         // 2. 添加到可视化对象列表
         addChild( loader );
         // 3. 调用load(    )方法
         loader.load( new URLRequest( "image.jpg" ) );
      }
   }
}
```

当运行代码时，播放器寻找*.swf* 文件所在目录下的图片文件，*URLRequest* 对象使用相对URL，也可以是绝对URL。如果下载成功会自动当作*Loader* 实例的子对象。

在载入外部资源的过程中可能会出现错误，比如，可能是URL地址不正确，或者是Flash播放器安全沙漏不允许，或者资源太大需要很长的时间下载，需要一个进度条告诉用户下载进度。

面对这些可能的情况，我们需要添加一个事件去检测它，*Loader*实例的*contentLoaderInfo* 属性会对不同的情况作出不同的反应事件。*contentLoaderInfo* 属性是*flash.display.LoaderInfo* 类实例，用来提供目标被载入时的信息，下面是*LoaderInfo* 类的一些有用的事件：

open

当资源开始下载时触发

progress

资源在下载中时触发

complete

当资源下载完成时触发

init

当载入外部的.swf初始化时触发

httpStatus

当载入外部资源的HTTP请求产生状态代码错误时触发

ioError

当一个错误导致下载被终止时触发，比如找不到相应资源

securityError

当试图读取安全沙漏以外的数据时触发

unload

当*unload()* 方法被调用或移除载入的内容时或再次调用*load()* 方法时都会触发该事件

下面的例子演示了contentLoaderInfo.的事件：

```
package {
   import flash.display.*;
   import flash.text.*;
   import flash.net.URLRequest;
   import flash.events.*;
   public class LoaderExample extends Sprite {
      public function LoaderExample(    ) {
         // Create the loader and add it to the display list
```

```
            var loader:Loader = new Loader(    );
            addChild( loader );
            loader.contentLoaderInfo.addEventListener( Event.OPEN, handleOpen );
            loader.contentLoaderInfo.addEventListener( ProgressEvent.PROGRESS, handleProgress );
            loader.contentLoaderInfo.addEventListener( Event.COMPLETE, handleComplete );
            loader.load( new URLRequest( "image.jpg" ) );
        }
        private function handleOpen( event:Event ):void {
            trace( "open" );
        }
        private function handleProgress( event:ProgressEvent ):void {
            var percent:Number = event.bytesLoaded / event.bytesTotal * 100;
            trace( "progress, percent = " + percent );
        }
        private function handleComplete( event:Event ):void {
            trace( "complete" );
        }
    }
}
```

运行上面的代码，在控制台上会显示下载百分比等信息。

下面我们修改下代码，在下载的过程中显示百分比。比如*handleOpen()* 方法创建一个显示百分比的文本框，*handleProgress()* 更新文本框的百分比，最后*handleComplete()* 方法进行清除，因为资源已全部下载：

```
private function handleOpen( event:Event ):void {
    _loaderStatus = new TextField(    );
    addChild( loaderStatus );
    _loaderStatus.text = "Loading: 0%";
}
private function handleProgress( event:ProgressEvent ):void {
    var percent:Number = event.bytesLoaded / event.bytesTotal * 100;
    _loaderStatus.text = "Loading: " + percent + "%";
}
```

```
private function handleComplete( event:Event ):void {
    removeChild( loaderStatus );
    _loaderStatus = null;
}
```

在类中添加_loaderStatus 变量，类型为*TextField*。

```
private var _loaderStatus:TextField;
```

6.7. 载入外部swf文件并与之交互

问题

我要载入而且控制外部的SWF文件

解决办法

使用*Loader* 类载入*.swf* 文件，然后通过*Loader*实例的content 属性访问

讨论

6.6节 演示了如何通过*Loader* 类载入外部图片资源，载入*swf* 文件也是同样的方法，通过*load()* 方法，然后传递一个.swf文件的URL。

我们创建两个独立的.swf文件：*ExternalMovie.swf* 和*LoaderExample.swf*。第一个稍后要被载入到第二个上，也就是*LoaderExample.swf*。*ExternalMovie.swf* 的代码如下：

```
package {
    import flash.display.Sprite;
    import flash.display.Shape;
    public class ExternalMovie extends Sprite {
        private var _color:uint = 0x000000;
        private var _circle:Shape;
        public function ExternalMovie(    ) {
            updateDisplay(    );
        }
        private function updateDisplay(    ):void {
            // 如果circle 没有创建则创建之并显示
            if ( _circle == null ) {
                _circle = new Shape(    );
```

```
            addChild( _circle );
        }
        // 清除以前画的内容重新画并填充之
        _circle.graphics.clear(    );
        _circle.graphics.beginFill( _color );
        _circle.graphics.drawCircle( 100, 100, 40 );
    }
    // 改变颜色
    public function setColor( color:uint ):void {
        _color = color;
        updateDisplay(    );
    }
    // 获取颜色
    public function getColor(    ):uint {
        return _color;
    }
  }
}
```

ExternalMovie.swf 运行时就画了一个黑色的圆，需要注意的是有两个用来访问和设置颜色的方法*getColor()* 和*setColor()*。无论何时*setColor()* 方法被调用，圆立即被重画。

申明方法为public，这样载入它的类也可以访问这些方法，*updateDisplay()* 方法是私有的，也就是说*LoaderExample.swf*是无法调用它的。

现在通过*LoaderExample.swf*去载入*ExternalMovie.swf*，代码如下：

```
package {
  import flash.display.*;
  import flash.net.URLRequest;
  import flash.events.Event;
  public class LoaderExample extends Sprite {
    private var _loader:Loader;
    public function LoaderExample(    ) {
      // 创建Loader并显示
      _loader = new Loader(    );
```

```
            addChild( _loader );
            // 添加于载入swf文件的交互事件
            _loader.contentLoaderInfo.addEventListener( Event.INIT, handleInit );
            // 载入外部文件
            _loader.load( new URLRequest( "ExternalMovie.swf" ) );
        }
        // 当文件载入完时触发该函数
        private function handleInit( event:Event ):void {
            // 这里设置为* 就是指事先不知道载入什么类型
            var movie:* = _loader.content;
            // 获得颜色值，显示为0
            trace( movie.getColor(    ) );
            // 设置颜色
            movie.setColor( 0xFF0000 );
        }
    }
}
```

LoaderExample.swf 的代码完成了与被载入swf进行交互，有两个重要步骤：

监听init 事件

通过content属性访问载入的swf。

当外部的swf载入完成并进行初始化时就会激活 init 事件，只有激活了该事件之后才可控制，否则就会出现运行时错误。

要控制外部载入的swf文件，必须先得到它的引用，这可通过*Loader*类的content属性获得。上面的例子中loader.content指向外部swf引用。如果loader 变量不可用，还可通过event.target.content得到*Loader*的content属性。

content 属性是只读的，返回*DisplayObject*类型对象。在*LoaderExample.swf* 代码中，movie变量类型为*，实际上也可以指定为*DisplayObject*。

*ExternalMovie.swf*被载入后，它有两个public 方法可以使用，如果调用不存在的方法则会抛出*ReferenceError* 异常。

getColor() 方法返回 *ExternalMovie.swf*中园的颜色，The *setColor()* 方法改变园的颜色，这里我们设置为红色。

6.8. 接收鼠标事件

问题

我要让动画接收鼠标命令

解决办法

使用各种鼠标事件监听鼠标动作，并作出反应。使用*DisplayObject* 实例的只读属性mouseX 和mouseY检查鼠标相对位置，或者*MouseEvent*事件的localX 和localY 属性。

讨论

6.5节 讨论的*SimpleButton* 类已经实现了最基本的鼠标交互。*SimpleButton* 类提供了简单的方法利用不同的对象状态创建鼠标交互按钮。

但是鼠标事件不仅仅只有这些，监听各种鼠标事件可以创建更加丰富的交互体验。比如你想在绘图程序中跟踪鼠标轨迹，随着用户鼠标的移动在屏幕上绘制线条，或者有一个迷宫程序，需要用鼠标做导航来走出迷宫，或者更多的复杂情况。

这些情况都需要使用特殊的*InteractiveObject* 可视化对象，它提供了响应用户鼠标的能力。如果回过头来看那张可视化对象层级图，你会发现*InteractiveObject* 类处在很高的层上，那些*Sprite*, *Loader*, *TextField*, 和*MovieClip* 类都在它的下面。

InteractiveObject 实例发出必要的鼠标事件，下面是些常用的鼠标事件：

click

当用户在显示的物体上单击鼠标时触发。

doubleClick

当用户在显示物体上双击鼠标时触发。

mouseDown

当用户在显示物体上按下鼠标键时触发。

mouseUp

当用户松开鼠标时触发

mouseOver

当用户移动鼠标到物体上时触发

mouseMove

当用户在物体上移动鼠标触发。

mouseOut

当用户移动鼠标离开物体时触发。

mouseWheel

当用户在物体上使用鼠标滚轮时触发。

使用*InteractiveObject*的*addEventListener()* 方法注册鼠标事件并定义*MouseEvent*事件处理函数。

下面的代码创建了一个*Sprite*, 画了个红色的圆，当鼠标移动到园上时在控制台输出信息。

```
package {
   import flash.display.Sprite;
   import flash.events.*;
   import flash.geom.Point;
   public class InteractiveMouseDemo extends Sprite {
      public function InteractiveMouseDemo(    ) {
         var circle:Sprite = new Sprite(    );
         circle.x = 10;
         circle.y = 10;
         circle.graphics.beginFill( 0xFF0000 );
         circle.graphics.drawCircle( 0, 0, 5 );
         circle.graphics.endFill(    );
         circle.addEventListener( MouseEvent.MOUSE_MOVE, handleMouseMove );
         addChild( circle );
      }
      private function handleMouseMove( event:MouseEvent ):void {
         trace( "mouse move" );
      }
   }
}
```

我们看到当鼠标在园上移动时，控制台上不同的输出“mouse move”。

另一个经常用的鼠标事件，就是鼠标的定位。比如说用鼠标画一条线，需要知道在哪里画线，也就是要得到鼠标的位置。这里有两种方法得到鼠标的位置：

使用*DisplayObject*实例的mouseX 和mouseY 属性

通过鼠标事件中*MouseEvent* 实例localX 和localY 属性。

mouseX 和mouseY 属性指示鼠标相对于*DisplayObject*对象左上角的坐标。这两个属性是只读的，不能被设置，只能检查鼠标位置。

想象一下如果有个矩形在x坐标为20 和y坐标为50 的地方，而鼠标的位置为x坐标25和y坐标60。那么矩形的mouseX 属性返回5，mouseY属性返回10。

MouseEvent 的localX 和localY 属性也是相对值。它是相对于激活事件的对象位置。当矩形返

回mouseX 为10时发出mouseMove 事件，事件的localX 属性也是10。

如果想从本地坐标中获得全局坐标，可用*DisplayObject* 类的*localToGlobal( )* 方法。*localToGlobal( )* 方法以*flash.geom.Point* 作为参数，返回新的*Point* 对象。下面的代码演示了事件处理和如何把localX 和localY 转换为全局坐标：

```
// Event handler to respond to a mouseMove event
private function handleMouseMove( event:MouseEvent ):void {
   /* 显示:
   local x: 3.95
   local y: 3.45
   */
   trace( "local x: " + event.localX );
   trace( "local y: " + event.localY );
   // 创建point
   var localPoint:Point = new Point( event.localX, event.localY );
   // 得到全局坐标
   var globalPoint:Point = event.target.localToGlobal( localPoint );
   /* 显示:
   global x: 13.95
   global y: 13.45
   */
   trace( "global x: " + globalPoint.x );
   trace( "global y: " + globalPoint.y );
}
```


下面的完整例子通过一个绘图程序演示了各种鼠标事件，无论何时鼠标按下，绘图将立即开始，当移动鼠标，一条线跟着鼠标显示出来，当松开鼠标，绘图终止：

```
package {
   import flash.display.Sprite;
   import flash.events.MouseEvent;
   public class DrawingDemo extends Sprite {
      // Flag to indicate whether the mouse is in draw mode
      private var _drawing:Boolean;
      public function DrawingDemo(    ) {
```

```
            // 设置线条样式
            graphics.lineStyle( 2, 0xFFCC33 );
            // 按住鼠标就绘图
            _drawing = false;
            // 监听鼠标事件
            stage.addEventListener( MouseEvent.MOUSE_DOWN, startDrawing );
            stage.addEventListener( MouseEvent.MOUSE_MOVE, draw );
            stage.addEventListener( MouseEvent.MOUSE_UP, stopDrawing );
        }
        public function startDrawing( event:MouseEvent ):void {
            // 绘制起点
            graphics.moveTo( mouseX, mouseY );
            _drawing = true;
        }
        public function draw( event:MouseEvent ):void {
            if ( _drawing ) {
                // 鼠标走到哪画到哪
                graphics.lineTo( mouseX, mouseY );
            }
        }
        public function stopDrawing( event:MouseEvent ):void {
            _drawing = false;
        }

    }
}
```

6.9. 用鼠标拖拽对象

问题

我想要用鼠标随意拖动对象

解决办法

使用 *Sprite* 的 *startDrop( ), stopDrag( )* 和 dropTarget 实现拖拽效果。另外也可以继承 *ascb.display.DraggableSprite* 类，使用*drag( )*和*drop( )* 方法。

讨论

创建拖动效果并不是想象的那么困难。*Sprite* 类包括了这些拖动的方法，分别是*startDrag( )* 和 *stopDrag( )*。

startDrag( ) 方法可在任何*Sprite* 实例上调用，要想停止拖动则调用*stopDrag( )* 方法，如果拖动完成，可以检查*Sprite*的dropTarget 属性是否是目标位置。

不需要指定任何参数就可调用*startDrag( )*，但实际上该方法有两个参数：

lockCenter

当为TRue 时*Sprite*的中心被鼠标位置锁定，不管鼠标是否按下。当为false 时只有鼠标点击它时 *Sprite* 才会跟着移动，默认为false。

bounds

一个矩形范围来约束拖动，被拖动*Sprite*是不能超出这个范围，默认为null，意味着没有约束。

下面的代码使用这些方法建立了简单的拖动效果。有三个不同颜色的矩形，右边有个白色的矩形作为拖动的目标，当拖动左边的矩形到白色矩形上时，松开鼠标，白色矩形就会i改变相应颜色。:

```
package {
   import flash.display.Sprite;
   import flash.display.DisplayObject;
   import flash.events.MouseEvent;
   import flash.geom.Point;
   import flash.filters.DropShadowFilter;
   public class ColorDrop extends Sprite {
      private var _red:Sprite;
      private var _green:Sprite;
      private var _blue:Sprite;
      private var _white:Sprite;
```

```
// 保存被拖动对象原始坐标
private var startingLocation:Point;
// 创建矩形和添加事件
public function ColorDrop(    ) {
    createRectangles(    );
    addEventListeners(    );
}
private function createRectangles(    ):void {
    _red = new Sprite(    );
    _red.graphics.beginFill( 0xFF0000 );
    _red.graphics.drawRect( 0, 10, 10, 10 );
    _red.graphics.endFill(    );
    _green = new Sprite(    )
    _green.graphics.beginFill( 0x00FF00 );
    _green.graphics.drawRect( 0, 30, 10, 10 );
    _green.graphics.endFill(    );
    _blue = new Sprite(    );
    _blue.graphics.beginFill( 0x0000FF );
    _blue.graphics.drawRect( 0, 50, 10, 10 );
    _blue.graphics.endFill(    );
    _white = new Sprite(    );
    _white.graphics.beginFill( 0xFFFFFF );
    _white.graphics.drawRect( 20, 10, 50, 50 );
    _white.graphics.endFill(    );
    addChild( _red );
    addChild( _green );
    addChild( _blue );
    addChild( _white );
}
private function addEventListeners(    ):void {
    _red.addEventListener( MouseEvent.MOUSE_DOWN, pickup );
```

```
    _red.addEventListener( MouseEvent.MOUSE_UP, place );
    _green.addEventListener( MouseEvent.MOUSE_DOWN, pickup );
    _green.addEventListener( MouseEvent.MOUSE_UP, place );
    _blue.addEventListener( MouseEvent.MOUSE_DOWN, pickup );
    _blue.addEventListener( MouseEvent.MOUSE_UP, place );
}
public function pickup( event:MouseEvent ):void {
    // 保存原始坐标以便回位
    startingLocation = new Point(    );
    startingLocation.x = event.target.x;
    startingLocation.y = event.target.y;
    //开始拖动，给被拖动对象加上阴影
    event.target.startDrag(    );
    event.target.filters = [ new DropShadowFilter(    ) ];
    // 把被拖动对象显示在最前面
    setChildIndex( DisplayObject( event.target ), numChildren - 1 );
}
public function place( event:MouseEvent ):void {
    // 停止拖动，取消阴影效果
    event.target.stopDrag(    );
    event.target.filters = null;
    // 检测是否已经被拖动到白色矩形上
    if ( event.target.dropTarget == _white ) {
        // 设置颜色
        var color:uint;
        switch ( event.target ) {
            case _red: color = 0xFF0000; break;
            case _green: color = 0x00FF00; break;
            case _blue: color = 0x0000FF; break;
        }
        _white.graphics.clear(    );
```

```
            _white.graphics.beginFill( color );
            _white.graphics.drawRect( 20, 10, 50, 50 );
            _white.graphics.endFill(    );
        }
        // 把被拖动对象放回原位
        event.target.x = startingLocation.x;
        event.target.y = startingLocation.y;
    }
  }
}
```

因为左边三个对象上都注册了mouseDown 和mouseUp 事件监听器，每次在相应对象上按下鼠标鼠标都会执行pickup 函数。

首先，保存被拖动对象的原始坐标，这样拖动完就能恢复位置。接着*startDrag()* 方法被调用开始拖动，而且加上了*DropShadowFilter* 效果，并且通过*setChildIndex()* 把被拖动对象设为最顶层。

当松开鼠标后mouseUp事件被激活，就会执行*place()* 方法。首先，调用*stopDrag()* 停止拖动，阴影效果被取消，接着检测拖动的目标是否为白色矩形，如果是就把被拖动对象的颜色应用在白色矩形上。然后恢复被拖动对象到原始位置上。

上面的代码看起来很好，但还是有两个小问题，那就是dropTarget 属性不总是可靠和稳定的。

在鼠标移动的过程中dropTarget 属性一直在不停的改变。这是对的，这样我们就能为不同的目标分别处理。并且可以分辨哪些目标才是我们需要的。但是dropTarget 的改变只有在鼠标点移动到新的物体上时才发生，也就是说当鼠标点离开物体它是不发生变化的。这就会有问题，比如当我们移动到一个物体上然后离开，但是并没有移动新的物体上，这时dropTarget 属性仍然指向原来的物体而不管鼠标已经离开物体了，这样就产生了偏差，请看下面：

看下面的情况：

捡起红色方框
移动到白色方框上
再移出白色方框
释放红色方框

我们看到白色方框的颜色还是变为了红色，这时因为dropTarget 仍然指向白色方框，虽然红色方框已经离开白色方框范围。

要修复这个问题，可使用*hitTestPoint()* 方法决定鼠标的位置是否在dropTarget 目标范围之内。*hitTestPoint()* 方法接受x和y坐标，返回true或false。如果坐标在对象范围之内，还有第三个布尔型参数指定边界检测模式，false代表边界模式为矩形，true代表边界模式对象自身边界，默认为

false。

我们在*place()* 方法的判断语句中增加*hitTestPoint()*方法来检测dropTarget，来确认鼠标仍在白色方框范围之内：

```
if ( event.target.dropTarget == _white
   && _white.hitTestPoint( _white.mouseX, _white.mouseY ) ) {
```

另一问题是当鼠标移动时屏幕更新会变得不稳定。这是因为鼠标事件不依赖于渲染处理。Flash帧数决定何时屏幕被刷新，因此如果鼠标改变了显示对象，但显示对象不会立即被更新，除非下一个帧到来（帧速决定快慢）。

为了解决这个问题，*MouseEvent* 类提供一个方法*updateAfterEvent()*。当mouseMove 事件触发，*updateAfterEvent()* 同志Flash 播放器屏幕已改变指示必须重画，这样就解决了帧速所造成的延时问题。

但不幸的是*updateAfterEvent()* 和*startDrag()*使用时效果总是不好。即使mouseMove 事件处理中调用了*updateAfterEvent()*促使渲染器更新，但是调用*updateAfterEvent()*后却没有效果。另一个我问题*startDrag()* 一次只能拖动一个*Sprite*。虽然这不是大问题，但也是一个限制。

为了处理以上问题，本书自定义了一个类叫*DraggableSprite*，在*ascb.display* 包中。

DraggableSprite 类继承自*Sprite*，提供了*drag()*和*drop()*。*drag()*方法和*startDrag()*一样接受同样的参数，使用方法也一样，*drop()* 方法和*stopDrag()*一样。

*DraggableSprite*最大的不同是实现了自定义鼠标跟踪代码，多个*DraggableSprite* 实例可同时拖动，而且*updateAfterEvent()* 也解决了渲染延时问题。

但是在*DraggableSprite*里dropTarget 属性不能用了，代替它的是*getObjectsUnderPoint()* 方法，返回鼠标位置下面的对象。

getObjectsUnderPoint() 方法返回一个对象数组，在数组的length-1位置上的对象就是最底层的对象，0位置是最上层的对象，根据这个数组然后检测白色方框是否在内就知道拖动目标是否合法了。

下面的代码和以前的基本一致，只不过*Sprite*换成了*DraggableSprite*：

```
package {
   import flash.display.Sprite;
   import flash.display.DisplayObject;
   import flash.events.MouseEvent;
   import flash.geom.Point;
   import flash.filters.DropShadowFilter;
   import ascb.display.DraggableSprite;
   public class ColorDrop extends Sprite {
      private var _red:DraggableSprite;
```

```
private var _green:DraggableSprite;
private var _blue:DraggableSprite;
private var _white:Sprite;
// Saves the starting coordinates of a dragging Sprite so
// it can be placed back
private var startingLocation:Point;
// Create the rectangles that comprise the interface
// and wire the mouse events to make them interactive
public function ColorDrop(    ) {
    createRectangles(    );
    addEventListeners(    );
}
private function createRectangles(    ):void {
    _red = new DraggableSprite(    );
    _red.graphics.beginFill( 0xFF0000 );
    _red.graphics.drawRect( 0, 10, 10, 10 );
    _red.graphics.endFill(    );
    _green = new DraggableSprite(    )
    _green.graphics.beginFill( 0x00FF00 );
    _green.graphics.drawRect( 0, 30, 10, 10 );
    _green.graphics.endFill(    );
    _blue = new DraggableSprite(    );
    _blue.graphics.beginFill( 0x0000FF );
    _blue.graphics.drawRect( 0, 50, 10, 10 );
    _blue.graphics.endFill(    );
    _white = new DraggableSprite(    );
    _white.graphics.beginFill( 0xFFFFFF );
    _white.graphics.drawRect( 20, 10, 50, 50 );
    _white.graphics.endFill(    );
    addChild( _red );
    addChild( _green );
```

```
        addChild( _blue );
        addChild( _white );
    }
    private function addEventListeners(    ):void {
        _red.addEventListener( MouseEvent.MOUSE_DOWN, pickup );
        _red.addEventListener( MouseEvent.MOUSE_UP, place );
        _green.addEventListener( MouseEvent.MOUSE_DOWN, pickup );
        _green.addEventListener( MouseEvent.MOUSE_UP, place );
        _blue.addEventListener( MouseEvent.MOUSE_DOWN, pickup );
        _blue.addEventListener( MouseEvent.MOUSE_UP, place );
    }
    public function pickup( event:MouseEvent ):void {
        // Save the original location so you can put the target back
        startingLocation = new Point(    );
        startingLocation.x = event.target.x;
        startingLocation.y = event.target.y;
        // Start dragging the Sprite that was clicked on and apply
        // a drop shadow filter to give it depth
        event.target.drag(    );
        event.target.filters = [ new DropShadowFilter(    ) ];
        // Bring the target to front of the display list so
        // that it appears on top of everything else
        setChildIndex( DisplayObject( event.target ), numChildren - 1 );
    }
    public function place( event:MouseEvent ):void {
        // Stop dragging the Sprite around and remove the depth
        // effect from the filter
        event.target.drop(    );
        event.target.filters = null;
        // Get a list of objects inside this container that are
        // underneath the mouse
```

```
            var dropTargets:Array = getObjectsUnderPoint( new Point( mouseX, mouseY ) );
            // The display object at position length - 1 is the top-most object,
            // which is the rectangle that is currently being moved by the mouse.
            // If the white rectangle is the one immedialy beneath that, the
            // drop is valid
            if ( dropTargets[ dropTargets.length - 2 ] == _white ) {
                // Determine which color was dropped, and apply that color
                // to the white rectangle
                var color:uint;
                switch ( event.target ) {
                    case _red: color = 0xFF0000; break;
                    case _green: color = 0x00FF00; break;
                    case _blue: color = 0x0000FF; break;
                }
                _white.graphics.clear(    );
                _white.graphics.beginFill( color );
                _white.graphics.drawRect( 20, 10, 50, 50 );
                _white.graphics.endFill(    );
            }
            // Place the dragging Sprite back to its original location
            event.target.x = startingLocation.x;
            event.target.y = startingLocation.y;
        }

    }
}
```

7.0. 介绍

在ActionScript中，我们可以通过编程画出*Shape*, *Sprite*, *Button*, 和*MovieClip*。每个类都有个graphics 属性，它是*flash.display.Graphics* 类实例。*Graphics* 类定义了一些绘图内容的API。这一章的讨论的基本上是如何使用*Graphics*类API。

因为*Shape*, *Sprite*, *Button*, 和*MovieClip* 类已经定义了graphics 属性，它就是*Graphics*实例的引用，所以没有必要再构造新的*Graphics* 对象。对象的graphics 属性可在对象内绘图。比如下面的代码设置了sprite的线条样式：

```
sampleSprite.graphics.lineStyle(    );
```

Graphics 类定义了一些基本的绘图方法，如直线，简单的图形。但是大多数的图形用*Graphics* API 还是很难画出的， AS3CBLibrary (http://www.rightactionscript.com/ascb) 提供了一个*ascb.drawing.Pen* 类。*Pen* 类是*Graphics* 类的代理（包装）类。你可以构造一个新的Pen 实例然后传递进*Graphics*对象引用作为参数：

```
var pen:Pen = new Pen(sampleSprite.graphics);
```

Pen 类代理了所有*Graphics*类的方法。这意味着*Graphics*的所有方法都可以在*Pen* 类中使用。另外*Pen* 类还定义了一些API 能更简单的画出弧线，椭圆，多边形，星形等等。我们会在下面的章节中讨论*Pen*类。

7.1. 设置线条样式

问题

我要设置线条样式

解决办法

使用*lineStyle()*方法

讨论

在绘画之前，必须先设置*Graphics*对象的线条样式。如果你没设置，默认的线条样式为undefined而且线条和填充都不能被渲染。可使用Graphics对象的*lineStyle()*方法设置。

*lineStyle()*方法接受多个参数，所以得参数都是可选的，他们都是：

thickness

定义线条的宽度，默认值为1，范围为0到255。如果指定的值超出它会自动工调整最接近的合法数值。

color

线条的颜色，默认为0x000000。

alpha

线条的透明度，范围为0到1，默认为1。

pixelHinting

布尔型，指示线条是否包住整个像素，默认为false。

scaleMode

*flash.display.LineScaleMode*中的一个常量，有NORMAL（默认), NONE, VERTICAL, 和HORIZONTAL。当值为NORMAL，随着所在对象的缩放而缩放。比如，如果有个含有1像素的线条的sprite被缩放到200%，那么线条的宽度也缩放到2个像素。如果设置为NONE，线条不会被缩放。如果设置为VERTICAL，则只在垂直方向所放，如果设为HORIZONTAL，则在水平方向所放。

caps

指定线条末端的封盖，它是*flash.display.CapsStyle* 的常量，有NONE, ROUND (默认)，和SQUARE.

joints

当连接线段时所指定的类型，它是*flash.display.JointStyle* 的常量，有BEVEL, MITER, 和ROUND (默认).

miterLimit

当连接类型为MITER 时还要指定斜接限制。默认为3，范围从1到255。

所以的参数都是可选的，像下面的代码这样是最简单的调用：

sampleSprite.graphics.lineStyle(　);

可以在任何需要的时候设置*lineStyle()*方法，比如说，你设置20像素画了绿线，再设置10像素画蓝线。
如果调用了 *clear()*方法后必须重新设置线条样式，不然就是 undefined 状态。

7.2. 设置渐变线条样式

问题

我要画出渐变样式的线条

解决办法

使用*Graphics.lineGradientStyle()*方法

讨论

*lineGradientStyle()*方法允许你画出渐变色的线条。如果已经设置了基本的线条样式，可以调用*lineGradientStyle()*应用渐变样式。它所需要的参数和 *beginGradientFill()*方法一样。

7.3. 画一条线

问题

我要用ActionScript画一条线。

解决办法

使用*Graphics.lineTo()*方法从当前位置到目标位置画一条线

讨论

画一条直线是最基本的绘图，Flash把鼠标当前位置作为起点，然后还需要个坐标作为目标点，使用*Grahics.lineTo()*方法从当前位置到目标位置创建一条直线。

```
// 从当前到(100,100) 画一条直线
sampleSprite.graphics.lineTo(100, 100);
```

当ActionScript 方法一调用，现实对象所关联的*Graphics* 对象就会画出相应的图形，比如上面的直线由sampleSprite所关联的*Graphics*画出。
*lineTo()*把笔刷当前位置作为起点，类似的还有*curveTo()*, 或*moveTo()* 方法。默认笔刷的起始坐标为(0,0)。*moveTo()* 方法不会画出东西，它直接把笔刷移动到目标位置：

```
// 在sampleSprite 里移动笔刷到(200,20)
sampleSprite.graphics.moveTo(200, 20);
```

当你需要设置笔刷的位置时*moveTo()* 方法是很有用的：

```
// 设置线条样式为1个像素，不透明
sampleSprite.graphics.lineStyle(    );
// 画出虚线
sampleSprite.graphics.lineTo(10, 0);
sampleSprite.graphics.moveTo(15, 0);
sampleSprite.graphics.lineTo(25, 0);
sampleSprite.graphics.moveTo(30, 0);
sampleSprite.graphics.lineTo(40, 0);
sampleSprite.graphics.moveTo(45, 0);
sampleSprite.graphics.lineTo(55, 0);
```

在7.1节, 讲过可以在任何时候重新设置线条样式，下面的代码画出四段不同颜色的线段：

```
// 设置线条样式为1像素，不透明
sampleSprite.graphics.lineStyle(    );
//画出虚线
```

```
sampleSprite.graphics.lineTo(10, 0);
sampleSprite.graphics.moveTo(15, 0);
//改变线条颜色为蓝色
sampleSprite.graphics.lineStyle(1, 0x0000FF);
sampleSprite.graphics.lineTo(25, 0);
sampleSprite.graphics.moveTo(30, 0);
//改变颜色为绿色
sampleSprite.graphics.lineStyle(1, 0x00FF00);
sampleSprite.graphics.lineTo(40, 0);
sampleSprite.graphics.moveTo(45, 0);
// 改变为红色
sampleSprite.graphics.lineStyle(1, 0xFF0000);
sampleSprite.graphics.lineTo(55, 0);
```

当你想在指定的位置画线时*Pen.drawLine()* 方法就很有用了。比如，下面的例子展示了如何画出虚线：

```
var pen:Pen = new Pen(sampleSprite.graphics);
pen.drawLine(0, 0, 10, 0);
pen.drawLine(15, 0, 25, 0);
pen.drawLine(30, 0, 40, 0);
pen.drawLine(45, 0, 55, 0);
```

7.4. 划一条曲线

问题

我要画一条曲线

解决办法

使用*Graphics.curveTo()* 方法

讨论

设置好了线条样式后，就可以使用*curveTo()* 方法画曲线了。*curveTo()* 方法画出接近于贝塞尔曲线（虽然是经过性能优化），它需要3点，开始点，控制点和目标点：

开始点一般是笔刷的当前位置。
指定曲线的目标终点。
控制点决定曲线的形状，它是根据起始点和目标点的切线计算出的。如果不指定控制点画出的就是直线：

*curveTo()*方法需要四个参数。前两个指定控制点的x，y坐标，后面两个指定目标点的x，y坐标，下面的例子画出了控制点在0,100 和目标点在100,100的曲线：

```
sampleSprite.graphics.lineStyle(    );
sampleSprite.graphics.curveTo(0, 100, 100, 100);
```

7.5. 画一条扇形

问题

我要画一条弧线

解决办法

使用*Pen.drawArc()*方法

讨论

扇形是圆的一部分，它比*curveTo()*方法要困难的多，不过使用*Pen.drawArc()*方法，你可以画出任意半径和长度的扇形，*drawArc()*方法接受以下参数：

x

扇形中心的x坐标（圆的中心）

y

扇形中心的y坐标

radius

扇形半径

arc

扇形度数，指定为角度

startingAngle

扇形开始角度，默认为0

radialLines

布尔值，指示是否画出扇形两端点到中心的直线，默认为false。

下面的例子画出了半径为50，角度为80，开始角度为20的扇形：

```
var pen:Pen = new Pen(graphics);
pen.drawArc(100, 100, 50, 80, 20, true);
```

7.6. 画一个矩形

问题

我要画一个矩形

解决办法

使用*Graphics.drawRect( )*方法画出直角矩形，*Graphics.drawRoundRect( )*或*Graphics.drawRoundRectComplex()*方法画出圆角矩形。

讨论

要画一个简单的矩形可用*lineTo()*方法画出四条线段：

```
//指定1像素，实心的黑线
sampleSprite.graphics.lineStyle(1, 0, 100);
// 画出四线段
sampleSprite.graphics.lineTo(100, 0);
sampleSprite.graphics.lineTo(100, 50);
sampleSprite.graphics.lineTo(0, 50);
sampleSprite.graphics.lineTo(0, 0);
```

你看，一个矩形就画好了，但是这样需要调用四次*lineTo()*，而且也很难画出圆角矩形来。

使用*Graphics.drawRect()*方法就简单多了，直接可以画出标准的矩形。该方法需要四个参数：左上角的x，y坐标和右下角的x，y坐标。下面的代码画了一个100x50 的矩形，左上角为0,0：

```
sampleSprite.graphics.lineStyle( );
sampleSprite.graphics.drawRect(0, 0, 100, 50);
```

*Graphics.drawRoundRect()*方法可画出圆角矩形,只不过后面多了个参数圆角半径，看下面的例子：

```
sampleSprite.graphics.lineStyle( );
sampleSprite.graphics.drawRoundRect(0, 0, 100, 50, 20);
```

*Graphics.drawRoundRectComplex()*基本上和*drawRoundRect()*一样，只不过它可以指定每个圆角的半径：

```
sampleSprite.graphics.lineStyle( );
sampleSprite.graphics.drawRectComplex(0, 0, 100, 50, 0, 20, 5, 25);
sampleSprite.graphics.endFill( );
```

7.7. 画一个圆

问题

我要画一个圆

解决办法

使用*Grahics.drawCircle()*方法

讨论

画圆就没有原始的方法可以用了，如果用理论算法画出每个像素点然后构成圆那太费时间了而且也不现实。还好Graphics类有个*drawCircle()*方法可以简单的画出圆，*drawCircle()*参数如下：

x

圆中心的x坐标

y

圆中心的y坐标

radius

圆半径

下面画出了中心点在100，100，半径为50的圆：

```
sampleSprite.graphics.lineStyle(    );
sampleSprite.graphics.drawCircle(100, 100, 50);
```

画一个同心圆：

```
sampleSprite.graphics.lineStyle(    );
sampleSprite.graphics.drawCircle(100, 100, 50);
sampleSprite.graphics.drawCircle(100, 100, 100);
```

也可用*beginFill()*, *beginGradientFill()*, 或*beginBitmapFill()* 填充圆，以*endFill()*结束：

```
sampleSprite.graphics.lineStyle(    );
sampleSprite.graphics.beginFill(0xFF0000);
sampleSprite.graphics.drawCircle(100, 100, 50);
sampleSprite.graphics.endFill(    );
```

7.8. 画一个椭圆

问题

我要画一个椭圆

解决办法

使用*Pen.drawEllipse()*方法

讨论

画椭圆比画圆就抽象多了，而且*Graphics*类没提供画椭圆的方法（原书错误，系统有提供的），使用*Pen*类的*drawEllipse()*方法是比较简单的方法，它需要四个参数：

x

椭圆中心的x坐标

y

椭圆中心的y坐标

xRadius

椭圆x方向的半径

yRadius

椭圆y方向的半径

下面的代码定义了*Pen*对象并画出一个椭圆：

```
var pen:Pen = new Pen(sampleSprite.graphics);
pen.drawEllipse(100, 100, 100, 50);
```

7.9. 画一个三角形

问题

我要画一个三角形

解决办法

使用*Pen.drawTriangle()*方法

讨论

给出两条边的长度和夹角就能确定一个三角形。*Pen*类的*drawTriangle()*方法就是基于此算法。它接受6个参数，如下：

x

夹角点的x坐标

y

夹角点的y坐标

ab

a（夹角点）到b的长度

ac

a（夹角点）到c的长度

angle

ab与ac的夹角

rotation

三角形的旋转角度。如果是0或undefined，那么ac平行于x轴。

一旦定义了*Pen*实例，就可以使用*drawTriangle()*方法快速的画出一个三角形，看下面的例子：

```
var pen:Pen = new Pen(sampleSprite.graphics);
pen.drawTriangle(100, 100, 100, 200, 40);
```

下面的代码填充三角形：

```
var pen:Pen = new Pen(sampleSprite.graphics);
pen.beginFill(0xFF0000);
pen.drawTriangle(100, 100, 100, 200, 40);
pen.endFill(    );
```

7.10. 画出规则的多边形

问题

我要画出规则的多边形（所有的边相等）

解决办法

使用*Pen.drawRegularPolygon()*方法

讨论

利用基本的三角数学也能画出规则的多边形，但是那样可能会花很多代码，用pen类的*drawRegularPolygon()*方法就很简单了。

drawRegularPolygon() 接受5个参数，如下：

x

多边形中心的x坐标

y

多边形中心的y坐标

sides

多边形边数

length

边长度

rotation

旋转角度

用下面的代码可以画出任意边数的多边形了（最少三条边）：

```
var pen:Pen = new Pen(sampleSprite.graphics);
// 画出长度为50的5边形
pen.drawRegularPolygon(100, 100, 5, 50);
```

下面是使用*drawRegularPolygon()*填充多边形：

```
var pen:Pen = new Pen(sampleSprite.graphics);
pen.beginFill(0xFF0000);
pen.drawRegularPolygon(100, 100, 5, 50);
pen.endFill(    );
```

7.11. 绘制星形

问题

我要画出星形

解决办法

使用*Pen.drawStar()*方法

讨论

*Pen.drawStar()*方法可以让你快速画出一个星形，它接受如下参数：

x

星形中心的x坐标

y

星形中心的y坐标

points

星形的顶点数

innerRadius

内半径

outerRadius

外半径

rotation

旋转角度，默认为0，

下面的例子画了个五顶点的星形：

```
var pen:Pen = new Pen(sampleSprite.graphics);
pen.drawStar(100, 100, 5, 50, 100);
```

7.12. 用实心和半透明来填充图形

问题

我想填充图形

解决办法

使用*Graphics.beginFill()*和*Graphics.endFill()*方法对完成图形填充

讨论

要填充图形，在画图形之前必须前调用*beginFill()*方法，画好图形后调用*endFill()*方法结束填充 。

你不能填充已经画好的图形，在填充图形之前必须前记得先调用*beginFill()*方法。
下面的例子创建了一个实心的绿方框：

```
sampleSprite.graphics.lineStyle(    );
sampleSprite.graphics.beginFill(0x00FF00);
sampleSprite.graphics.lineTo(100, 0);
sampleSprite.graphics.lineTo(100, 100);
sampleSprite.graphics.lineTo(0, 100);
sampleSprite.graphics.lineTo(0, 0);
sampleSprite.graphics.endFill(    );
```

*MovieClip.beginFill()*方法接受两个参数：
fillColor

指定RGB值作为填充色

alpha

值范围在0到1之间，用于控制透明度，默认为1。

要想创建透明的填充图形，*alpha*值小于1就行，如果*alpha*为0，就不会填充图形了。

*endFill()*方法不需要参数，它指示以 *beginFill()*, *beginGradientFill()*, 或 *beginBitmapFill()*.开始的填充完成。

7.13. 用渐变色填充图形

问题

我要用渐变色填充图形

解决办法

使用*beginGradientFill()*和*endFill()*方法完成渐变填充

讨论

渐变色填充是用多种颜色之间的渐变层次来做填充色。Flash支持线形渐变，就是左边的颜色渐变到右边的颜色（这是水平渐变，也可以垂直渐变），或者指定旋转角度。Flash也支持放射渐变，它是从中心到四周渐变颜色。通过*beginGradientFill()*启动渐变填充，它的参数有：：

gradientType

*flash.display.GradientType*类常量，为LINEAR或RADIAL。

colors

RGB 值的数组

alphas

透明度值数组

ratios

根据颜色和透明度指定最后的纯度，范围为0到255。

matrix

flash.geom.Matrix 对象定义用于渐变的转换。默认为1x1的渐变转换来填充图形。Matrix类定义了*createGradientBox()*方法，它接受以下参数：

scaleX

水平缩放比

scaleY

垂直缩放比

rotation

旋转角度，可以把角度转换为弧度，公式为*Math.PI/180*；默认为0

tx

x方向的转换数量，默认为0

ty

y方向的转换数量，默认为0

spreadMethod

flash.display.SpreadMethod 类常量，有PAD, REFLECT, 和REPEAT。默认为PAD。

interpolationMethod

*flash.display.InterpolationMethod*类常量，有LINEAR_RGB和RGB。默认为RGB。插补方法影响颜色渐变。

focalPointRatio

改值范围从-1 到1 指示渐变的焦点（对线形渐变无效）。默认为0，也就是位于中点。

下面的例子画出一个渐变填充圆：

```
var matrix:Matrix = new Matrix(    );
matrix.createGradientBox(100, 100, 0, 50, 50);
var colors:Array = [0xFF0000, 0x0000FF];
var alphas:Array = [100, 100];
var ratios:Array = [0x00, 0xFF];
sampleSprite.graphics.lineStyle(    );
sampleSprite.graphics.beginGradientFill(GradientType.GRADIENT, colors, alphas, ratios, matrix);
sampleSprite.graphics.drawCircle(100, 100, 50);
sampleSprite.graphics.endFill(    );
```


7.14. 用位图填充图形

问题

我要用位图填充图形

解决办法

使用*Graphics.beginBitmapFill()*方法

讨论

*Graphics.beginBitmapFill()*方法允许用位图填充图形，它接受以下参数：

bitmap

填充用的*BitmapData*对象

matrix

默认下不需要应用转换，也可指定*flash.geom.Matrix*对象进行位图的缩放，旋转，倾斜，透明等变换。

repeat

布尔值，指定是否平铺位图，默认为true。

smooth

布尔值，指示对位图进行光滑处理，默认为false。

下面的例子通过URL载入位图，拷贝到*BitmapData*对象上，使用*BitmapData*对象填充圆：

```
package {
   import flash.display.Sprite;
   import flash.geom.Matrix;
   import flash.display.Loader;
   import flash.net.URLRequest;
   import flash.display.BitmapData;
   import flash.events.Event;
   public class Drawing extends Sprite {
      private var _loader:Loader;
      public function Drawing(    ) {
         _loader = new Loader(    );
         _loader.load(new URLRequest("http://www.rightactionscript.com/samplefiles/image2.jpg"));
         _loader.contentLoaderInfo.addEventListener(Event.COMPELTE, onImageLoad);
      }
      private function onImageLoad(event:Event):void {
         var bitmap:BitmapData = new BitmapData(_loader.width, _loader.height);
         bitmap.draw(_loader, new Matrix(    ));
         var matrix:Matrix = new Matrix(    );
         matrix.scale(.1, .1);
         var sampleSprite:Sprite = new Sprite(    );
         sampleSprite.graphics.lineStyle(    );
         sampleSprite.graphics.beginBitmapFill(bitmap, matrix);
         sampleSprite.graphics.drawCircle(100, 100, 100);
         sampleSprite.graphics.endFill(    );
         addChild(sampleSprite);
      }
   } }
```

7.15. 编写遮罩

问题

我要创建遮罩

解决办法

使用*DisplayObject.mask*.

讨论

遮罩可用来创建唯一的图形或视觉效果。例如，可以创建擦除和过渡效果。还可以创建有趣的动画效果，甚至是图形扭曲效果。

任何可视化对象都可以作为任一对象的遮罩，下面的代码把*sampleSprite*的遮罩设置为maskSprite：

```
sampleSprite.mask = maskSprite;
```

下面的例子画了两个图形，一个作为遮罩。注意，两个对象都要通过*addChild()*方法添加到显示列表。虽然遮罩即使不添加到显示列表也能正常工作，不过还是建议你添加到显示列表：

```
var maskSprite:Sprite = new Sprite(    );
var pen:Pen = new Pen(maskSprite.graphics);
pen.beginFill(0xFFFFFF);
pen.drawArc(100, 100, 50, 80, 20, true);
pen.endFill(    );
var maskedSprite:Sprite = new Sprite(    );
maskedSprite.graphics.lineStyle(    );
maskedSprite.graphics.beginFill(0xFF0000);
maskedSprite.graphics.drawRect(0, 0, 200, 200);
maskedSprite.graphics.endFill(    );
maskedSprite.mask = maskSprite;
addChild(maskedSprite);
addChild(maskSprite);
```

下面的例子演示遮罩跟随鼠标移动。通过载入的图片指定遮罩，该遮罩跟着鼠标移动：

```
var loader:Loader = new Loader(    );
loader.load(new URLRequest("http://www.rightactionscript.com/samplefiles/image2.jpg"));
addChild(loader);
var maskSprite:Sprite = new Sprite(    );
```

```
maskSprite.graphics.lineStyle(    );
maskSprite.graphics.beginFill(0xFFFFFF);
maskSprite.graphics.drawCircle(0, 0, 50);
maskSprite.graphics.endFill(    );
loader.mask = maskSprite;
addChild(maskSprite);
maskSprite.startDrag(true);
```

8.0. 简介

和Flash 8 中的*BitmapData*类一样，这是非常重要的一个类，起初，Flash只是基于矢量的一个工具，矢量图形是由数学方法描述图形元素，比如一条直线是从*x0, y0* 扩展到*x1, y1*。 而一个位图，它把图形描述为一个矩形区域值集合，每个点都对应一个颜色值。

矢量图有两大优势：缩放和文件大小。当你缩放矢量图时，图像总能保持清晰，而位图当放大时就会出现锯齿状变得不清晰。

正因为矢量图是一个线条，曲线和图形的坐标组成的列表，所以它的文件和位图相比小很多。而位图要记录每个点的颜色值，信息量非常大，也就导致了文件的庞大。

矢量图的这种优势使它作为Flash媒体格式的最好形式，但是位图也有它自己的优点，比如位图能很好的表现照片，如果用矢量图去描述照片的所有颜色和图形，那结果就是比位图的文件还要大了。

位图另一个好处就是很容易进行处理，矢量图的每个曲线都是经过计算出来的，如果图像复杂就会花很长时间去处理，而位图就容易处理多了，无论多么复杂的动画，位图总能表现的很好。

Flash 8 对位图的支持还是很有限的，虽然能够载入位图显示，但不能作过多的处理。*BitmapData*类提供了很好的方法来操作位图。

8.1. 创建BitmapData对象

问题

我要在程序里创建位图

解决办法

使用*BitmapData*类构造一个新的*BitmapData*对象

讨论

*BitmapData*类表示一个由象素组成的位图，包含了很多内建的方法控制和处理图像。第一步先创建该类实例：

```
var bitmap:BitmapData = new BitmapData(width, height,transparent, fillColor);
```

该类在*flash.display*包中，width和height 参数指定位图的宽度和高度，下一个参数为布尔型，指定是否创建alpha通道，fillColor 参数决定背景颜色。

*width*和*height*是必须的，*transparent*和*fillColor*默认为true和0xFFFFFFFF。

fillColor 接受32-bit的颜色值，这意味着它支持alpha通道。当然，相应的*BitmapData*的transparent属性应为true。否则，所以颜色都是不透明的。

下面的例子创建了*BitmapData* 对象，初始化为透明的带alpha通道，背景色为0：

```
var bitmap:BitmapData = new BitmapData(100, 100, true, 0x00FFFFFF);
```

创建出 *BitmapData* 实例后就存在于内存中了，虽然现在就可以创建内容，不过在没加入到显示列表中之前它是不显示的。

8.2. 添加位图到可视化对象列表

问题

我要创建*BitmapData* 并显示它

解决办法

使用*BitmapData* 创建位图，并加入到可视化对象列表。

讨论

在ActionScript 3.0里要让对象可视，则必须加入到可视化对象列表中才行，然而*addChild()* 方法添加的对象必须是*flash.display.DisplayObject*的子类才行，而*BitmapData* 类继承自*Object*，所以不能直接加到列表中。

要加到可视化对象列表中，可使用*flash.display.Bitmap*类，它是*DisplayObject.*类的子类，实际上是*BitmapData*的一个包装类，允许*BitmapData*可被显示。

当用*Bitmap*构造函数创建实例时把*BitmapData*引用作为参数传递给它，然后调用*addChild()*把*Bitmap* 加入到显示列表，下面的例子创建了红色背景的*BitmapData* ：

```
var bitmap:BitmapData = new BitmapData(100, 100, true, 0xffff0000);
var image:Bitmap = new Bitmap(bitmap);
addChild(image);
```

8.3. 绘制可视化对象到位图上

问题

我想把Sprite或其他显示对象上的内容绘制到位图上

解决办法

用*BitmapData*类的*draw()*方法绘制内容

讨论

创建好的*BitmapData*，只是简单的黑色背景。你可能会把在sprite或其他可视化对象上已画好的图形内容画到位图上，用*draw()*方法就能做到，你只要把相关对象作为*draw()*的参数就可以了，也可以把*flash.geom.Matrix*类实例作为参数，*Matrix*类允许你对图形进行缩放，旋转，变换，或倾斜等操作。该参数是可选的，不过你又要使用参数，但又不想做什么，可指定为*null*。你还可以传递*ColorTransform* 对象作为参数，它能在绘图前修改颜色，下面的例子画了一个sprite到*BitmapData*上，名为_sprite，*BitmapData* 名为bitmap：

```
bitmap.draw(_sprite);
```

BitmapData 类有一些基本绘图方法，比如设置象素颜色，创建填充的矩形，或噪波函数，但是没有基本的函数如画直线，曲线等等，为了解决这问题，可以先在movie clip或sprite画好，在把对象画到位图上去。下面的例子创建了*BitmapData*和*Sprite*，先在sprite 上画出椭圆，再把sprite画到*BitmapData*上：

```
var bitmap:BitmapData = new BitmapData(100, 100,true, 0x00ffffff);
var sprite:Sprite = new Sprite(    );
sprite.graphics.beginFill(0xff0000, 100);
sprite.graphics.drawEllipse(0, 25, 100, 50);
sprite.graphics.endFill(    );
bitmap.draw(sprite);s
```

8.4. 载入外部图片到位图上

问题

我要载入外部图片作为*BitmapData*处理

解决办法

使用*flash.display.Loader* 类载入图片，当图片载入完成时，通过loader的*content* 属性property，它就是个*Bitmap*。访问*Bitmap*的*bitmapData* 属性就在访问载入的图片

讨论

通过*Loader*类载入外部位图。通过*URLRequest* 对象和位图的URL，监听loader的*complete*事件确定是否载入完成：

```
package {
    import flash.display.Sprite;
    import flash.display.Loader;
    import flash.events.Event;
    import flash.net.URLRequest;
    public class BitmapLoader extends Sprite {
        private var _loader:Loader = new Loader( );
        public function BitmapLoader( ) {
            _loader.contentLoaderInfo.addEventListener(Event.COMPLETE, onComplete);
            _loader.load(new URLRequest("image.jpg"));
        }
    }
}
    public function onComplete(event:Event):void {
        var image:Bitmap = Bitmap(_loader.content);
        var bitmap:BitmapData = image.bitmapData;
        addChild(image);
    }
```

首先得到loader的content 属性，它是个可视化对象表示内容被载入。如果你载入的是外部的*.swf*，它就是*MovieClip*类型。在这里载入的是位图，所以content 是一个*Bitmap*，类型转化为*Bitmap* 编译器不会抱错。

接着就可以访问包含位图内容的*BitmapData*了，你可以修改或载入新的图片。下面的例子在图

片上画了个白色的矩形：

```
public function onComplete(event:Event):void {
        var loadedImage:Bitmap = Bitmap(_loader.content);
        // Create a new Bitmap data and draw the
        // loaded image to it.
        var bitmap:BitmapData = new BitmapData(loadedImage.width,
                                               loadedImage.height,
                                               false, 0xffffffff);
        bitmap.draw(loadedImage, new Matrix(    ))
        // Create a new Bitmap using the BitmapData
        // and display it.
        var image:Bitmap = new Bitmap(bitmap);
        addChild(image);
        // Manipulate the pixels as you wish
        bitmap.fillRect(new Rectangle(0, 0, 50, 50), 0xffffffff);
}
```

8.5. 处理象素

问题

我要读取和设置位图的单个象素

解决办法

使用*BitmapData*类的*getPixel()*, *setPixel()*, *getPixel32()*,和*setPixel32()* 方法

讨论

在ActionScript 3.0里设置和读取象素是很简单的。要读取象素值只要指定象素 x，y坐标给*getPixel()*方法就可以了。设置象素，只要把坐标和颜色值传给*setPixel()*方法。

getPixel() 和*setPixel()*方法是专门为不透明的*BitmapData* 类实例所用，而*getPixel32()* 和*setPixel32()* 则支持alpha通道的位图。不透明的位图是24-bit, 红，绿，蓝各占8位，透明位图多一个8位的alpha通道，可以通过*BitmapData* 构造器指定是否为透明位图。

下面的代码创建了一个白色的，32位的*BitmapData*，而且设置了1000 个随机红色半透明的象素值：

```
var bitmap:BitmapData = new BitmapData(100, 100, true, 0xffffffff);
var image:Bitmap = new Bitmap(bitmap);
addChild(image);
for(var i:int = 0; i < 1000; i++) {
    bitmap.setPixel32(Math.round(Math.random(    ) * 100),
                      Math.round(Math.random(    ) * 100),
                      0x88ff0000);
}
```

如果在透明的*BitmapData*上使用*setPixel()*，那么alpha 通道会被设置100不透明，如果在不透明的图片上使用*setPixel32()*，那么alpha通道会被忽略，相当于没有alpha通道。通常，最好在正确的类型上使用正确的方法。

*getPixel*方法返回的数取决于具体的位图类型，它返回24或32位的数字。下面的类设置了初步的颜色，首先使用*BitmapData*的*noise()*方法产生一些随机的颜色，然后加入一个文本框，监听*enterFrame* 事件。每一帧得到鼠标所在位置的象素值转换为十六进制字符串：

```
package {
    import flash.display.Sprite;
    import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.text.TextField;
```

```
        import flash.events.Event;
        public class ColorChooser extends Sprite {
            private var _bitmap:BitmapData;
            private var _textfield:TextField;
            public function ColorChooser(    ) {
                _bitmap = new BitmapData(100, 100, false, 0xffffffff);
                var image:Bitmap = new Bitmap(_bitmap);
                addChild(image);
                _bitmap.noise(1000, 0, 255, 1|2|4, false);
                _textfield = new TextField(    );
                addChild(_textfield);
                _textfield.y = 100;
                addEventListener(Event.ENTER_FRAME, onEnterFrame);
            }
            public function onEnterFrame(event:Event):void {
                var colorVal:Number = _bitmap.getPixel(mouseX, mouseY)
                _textfield.text = "#" + colorVal.toString(16).toUpperCase(    );
            }
        }
}
```

8.6. 创建矩形填充

问题

我要用指定的颜色填充一个位图的矩形区域

解决办法

使用*BitmapData*类的*fillRect()*方法

讨论

BitmapData 类没有提供绘图方法，只有些填充方法，使用起来也是很简单的，只要传递一个矩形和颜色就可以了：

```
_bitmap.fillRect(rectangle, color);
```

该矩形必须是*flash.geom.Rectangle* 类的实例。通过它的构造函数创建一个实例：

```
var rect:Rectangle = new Rectangle(0, 0, 50, 100);
```

下面的代码创建了一个白色背景的位图，然后在中间画了个红色的矩形：

```
public function RectExample(    ) {
        _bitmap = new BitmapData(100, 100, false, 0xffffffff);
        var image:Bitmap = new Bitmap(_bitmap);
        addChild(image);
        _bitmap.fillRect(new Rectangle(25, 25, 50, 50), 0xffff0000);
}
```

注意 *fillRect()* 可以同时处理透明和不透明位图，如果对应不透明位图，只要指定颜色为 24 位即可。

8.7. 创建一个充溢填充

问题

我要填充一个大的不规则的区域

解决办法

使用*BitmapData*类的*floodFill()*方法

讨论

*floodFill()*方法和*setPixel()*方法语法一样，接受一个x，y坐标和颜色。

看下面的代码演示，首先创建一个位图和一些随机的方框，然后鼠标点击某个方框，就会用红色填充它：

```
package {
    import flash.display.Sprite;
    import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.events.MouseEvent;
    import flash.geom.Rectangle;
    public class FloodFillDemo extends Sprite {
        private var _bitmap:BitmapData;
        public function FloodFillDemo (    ) {
            var sprite:Sprite = new Sprite(    );
            addChild(sprite);
            _bitmap = new BitmapData(stage.stageWidth,
                                     stage.stageHeight,
                                     false, 0xffffffff);
            for(var i:int = 0; i < 20; i++) {
                _bitmap.fillRect(new Rectangle(
                                 Math.random(    ) * stage.stageWidth,
                                 Math.random(    ) * stage.stageHeight,
                                 50, 50), Math.random(    ) * 0xffffffff);
            }
            var image:Bitmap = new Bitmap(_bitmap);
```

```
            sprite.addChild(image);
            sprite.addEventListener(MouseEvent.MOUSE_DOWN, onMouseDown);
        }
        public function onMouseDown(event:MouseEvent):void {
            _bitmap.floodFill(mouseX, mouseY, 0xffff0000);
        }
    }
}
```

8.8. 拷贝象素

问题

我要拷贝*BitmapData*中的象素

解决办法

使用*BitmapData*的*copyPixels()*方法

讨论

*copyPixels()*方法的实现也很简单，只要得到象素值然后画到其他地方，非常类似于*draw()*方法。但是*copyPixels()*可控制拷贝象素的数量和目标。只要指定一个矩形区域和目标点即可：

```
bitmap.copyPixels(sourceBmp, srcRect, destPoint);
```

srcRect是*flash.geom.Rectangle*类实例，destPoint是*flash.geom.Point*类实例，它指定拷贝到目标位图的具体x，y坐标位置。

下面的例子演示如何从载入的位图中拷贝多个矩形区域到另一个*BitmapData*上：

```
package {
    import flash.display.Sprite;
    import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.display.Loader;
    import flash.net.URLRequest;
    import flash.events.Event;
    import flash.geom.Point;
```

```
    import flash.geom.Rectangle;
    public class AS3CB extends Sprite {
        private var _bitmap:BitmapData;
        private var _loader:Loader;
        public function AS3CB(    ) {
            _loader = new Loader(    );
            _loader.contentLoaderInfo.addEventListener(Event.COMPLETE, onLoad);
            _loader.load(new URLRequest("myphoto.jpg"));
            _bitmap = new BitmapData(stage.stageWidth,
                                     stage.stageHeight,
                                     false, 0xffffffff);
            var image:Bitmap = new Bitmap(_bitmap);
            addChild(image);
        }
        public function onLoad(event:Event):void {
            var loaderBmp:Bitmap = Bitmap(_loader.content);
            var w:Number = loaderBmp.width / 10;
            for(var i:int = 0; i < 10; i++) {
                _bitmap.copyPixels(loaderBmp.bitmapData,
                                   new Rectangle(i * w, 0,
                                                 w, loaderBmp.height),
                                   new Point(i * (w + 2), i));
            }
        }
    }
}
```

8.9.拷贝通道

问题

我要从*BitmapData*中拷贝出红色，绿色，蓝色或alpha通道出来

解决办法

使用*BitmapData*的*copyChannel()*方法

讨论

*copyChannel()*方法是另一个在两个位图之间交换数据的方法。它的前三个参数和*copyPixels()*方法一样，另外还有源通道和目标通道：

```
bitmap.copyPixels(sourceBmp, srcRect, destPoint,
                  srcChannel, destChannel);
```

这两个通道参数可以1，2，4或8中的整数，它们代表红色，绿色，蓝色和alpha通道，一次对应*BitmapDataChannel*类中的RED, GREEN, BLUE,和ALPHA常量。

你只要告诉方法从原始图像的什么通道拷贝到目标位图的哪个通道上，下面的代码拷贝载入位图的红色，绿色和蓝色通道到另一个位图上：

```
var loaderBmp:Bitmap = Bitmap(loader.content);
bitmap.copyChannel(loaderBmp.bitmapData,
                   loaderBmp.bitmapData.rect,
                   new Point(    ),
                   BitmapDataChannel.RED,
                   BitmapDataChannel.RED);
bitmap.copyChannel(loaderBmp.bitmapData,
                   loaderBmp.bitmapData.rect,
                   new Point(5, 5),
                   BitmapDataChannel.GREEN,
                   BitmapDataChannel.GREEN);
bitmap.copyChannel(loaderBmp.bitmapData,
                   loaderBmp.bitmapData.rect,
                   new Point(10, 10),
                   BitmapDataChannel.BLUE,
                   BitmapDataChannel.BLUE);
```

8.10. 创建噪波图案

问题

我要创建随机噪波图案

解决办法

使用*BitmapData*类的*noise()*方法

讨论

噪波图案是一些随机的象素值组成的图案，虽然这也能通过循环调用*setPixel()*方法做到，但是*noise()*方法更简单些且功能强大。

*noise()*创建随机杂乱的图案，它就像我们的电视当没有信号时出现的图案那样，但是如果把噪波和其他滤镜组合使用会得到很有意思的效果。

直接通过*BitmapData*调用*noise()*方法，参数有：

```
bitmap.noise(seed, low, high, channel, grayscale);
```

*seed*参数决定随机样式，它可以是任意值。如果用同样的seed调用两次方法，得到的图案将是一样的，因此要的到不同的图案必须使用不同的seed，可以用随机数产生它：

```
Math.random(    ) * 100000
```

low和high 参数决定每个象素的最大值和最小值，范围在0到255，设置越高图案约亮，越低越暗。

channel参数指定把噪波应用到哪个通道上，其值可以是1，2，4，8，或者用*BitmapDataChannel*类的RED, GREEN, BLUE, 和ALPHA。

grayscale 是一个布尔值，true表示随机值应用到三个通道上。

看下面的例子，首先创建位图，添加噪波并显示：

```
bitmap = new BitmapData(stage.stageWidth, stage.stageHeight,
                                false, 0xff000000);
bitmap.noise(1000, 0, 255, BitmapDataChannel.RED, false);
var image:Bitmap = new Bitmap(_bitmap);
addChild(image);
```

这样在红色通道创建了随机的图案，提高最低值可产生更亮的图案：

```
bitmap.noise(1000, 200, 255, BitmapDataChannel.RED, false);
```

设置最后的参数为true，可转换为灰度图：

```
bitmap.noise(1000, 200, 255, BitmapDataChannel.RED, true);
```

如果设置grayscale参数为true，不管你指定什么通道，都只能创建灰度颜色的噪点：

```
bitmap.noise(1000, 0, 255, BitmapDataChannel.RED |
                           BitmapDataChannel.GREEN |
                           BitmapDataChannel.BLUE,
                           false);
```

最后，我们试验下噪波和其他滤镜一起使用的效果如何。下面的代码创建了噪波，然后应用一个水平的模糊滤镜：

```
bitmap = new BitmapData(stage.stageWidth, stage.stageHeight,
                        false, 0xff000000);
bitmap.noise(1000, 128, 255, BitmapDataChannel.RED, true);
bitmap.applyFilter(bitmap,
                   bitmap.rect,
                   new Point(    ),
                   new BlurFilter(30, 1, 3));
var image:Bitmap = new Bitmap(bitmap);
addChild(image);
```

8.11. 创建布林噪波

问题

我要创建随机的类似天然的效果如云，烟或水

解决办法

使用*BitmapData*类的*perlinNoise()*方法

讨论

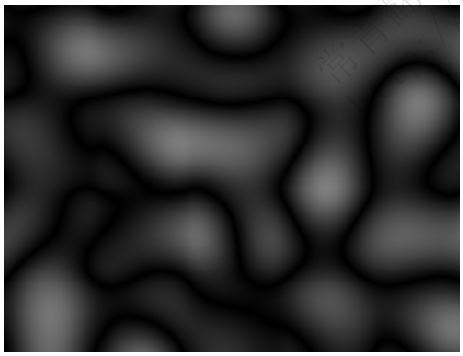
和*noise()*方法一样，*perlinNoise()*方法也创建随机图案，但是布林噪波的算法能产生类似自然图案，该算法由肯 布林发明，能产生如爆炸，烟雾，水等自然效果，因为它是基于算法的，其运算速度比创建同等位图快且占用内存少等优点，方法原型如下：

```
bitmap.perlinNoise(baseX, baseY, octaves, seed, stitch, fractal,
                   channels, grayscale, offsets);
```

前6个参数是必须的，后3个可选，现在我们创建一个简单的例子，下面的代码创建一个位图，应用布林噪波并显示：

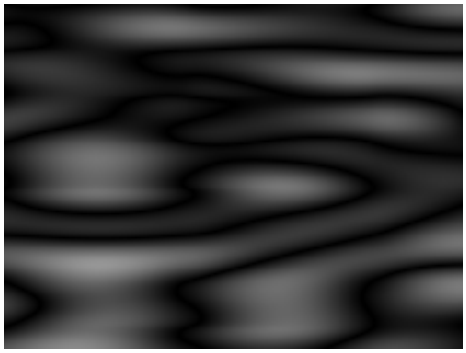
```
bitmap = new BitmapData(stage.stageWidth, stage.stageHeight, false, 0xff000000);
bitmap.perlinNoise(100, 100, 1, 1000, false, false, 1, true, null);
var image:Bitmap = new Bitmap(bitmap);
addChild(image);
```

效果如下：



试着修改下参数值看看效果如何。

baseX和baseY 决定图案的大小，这里设置成100，如果改成200和50，则会在水平上进行拉伸，效果如下：



octaves 参数是个整数，决定噪波的迭代次数，数值越大，产生越细的噪波，花费的时间也长一些。

*seed*参数和*noise()*方法中的一样意思，如果用同样的随机种子，产生的是同一个图案。

stitch参数为true时，图案四周相互协调，能使位图很小能平铺，看下面的代码：

```
bitmap = new BitmapData(100, 100, false, 0xff000000);
bitmap.perlinNoise(100, 100, 2, 1000, true, false, 1, true);
graphics.beginBitmapFill(bitmap);
graphics.drawRect(0, 0, stage.stageWidth, stage.stageHeight);
graphics.endFill(    );
```

ractal 参数为true时，图案被光滑且模糊化：

```
bitmap = new BitmapData(stage.stageWidth, stage.stageHeight,
                                false, 0xff000000);
bitmap.perlinNoise(200, 100, 5, 1000, false, false, 1, true, null);
var image:Bitmap = new Bitmap(bitmap);
addChild(image);
```

设置fractal为TRue:

```
bitmap.perlinNoise(200, 100, 5, 1000, false, true, 1, true, null);
```

这样的效果看起来像云

布林噪波还可建立在alpha通道上，这样可以创建透明的云彩或雾效果。

最后的参数表示偏移量，是个*Point* 对象数组，每个点指定单个分形的x，y坐标偏移量，如果布林噪波有多个分形（第三个参数代表分形），那么数组的长度就等于分形的数量，下面的例子创建了两个分形布林噪波图案在x轴上拉伸：

```
package {
    import flash.display.Sprite;
    import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.events.Event;
    import flash.geom.Point;
    public class Clouds extends Sprite {
        private var _bitmap:BitmapData;
        private var _xoffset:int = 0;
        public function Clouds(    ) {
            _bitmap = new BitmapData(stage.stageWidth, stage.stageHeight,
                                     true, 0xffffffff);
            var image:Bitmap = new Bitmap(_bitmap);
            addChild(image);
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
        }
        public function onEnterFrame(event:Event):void {
            _xoffset++;
            var point:Point = new Point(_xoffset, 0);
            // use the same point in both elements
            // of the offsets array
            _bitmap.perlinNoise(200, 100, 2, 1000, false, true,
                                1, true, [point, point]);
        }
    }
}
```

8.12. 使用阈值

问题

我要在位图上基于当前象素值进行修改

解决办法

使用*BitmapData*类的*threshold()*方法

讨论

*threshold()*方法可能是*BitmapData* 方法中最复杂的一个了，但是它的功能十分强大，该方法使用两个*BitmapData*对象：

destBitmap, 要修改的位图
sourceBitmap, 原位图，需要此位图象素作为计算数据

该方法用6个比较操作符和指定的值与源位图的每个象素作比较，如果比较结果为false，则目标位图相应位置的象素被设置为指定的值，如果比较为true，要么不改变，也可拷贝原位图的象素值。

这是方法原型：

```
destBitmap.threshold(sourceBitmap,
                     sourceRect,
                     destPoint,
                     operation,
                     threshold,
                     color,
                     mask,
                     copySource)
```

sourceRect是*flash.geom.Rectangle*类实例。它定义原位图要进行比较的区域，如果要比较整张位图，可传递sourceBitmap.rect 作为参数

destPoint 参数指定目标位图那些象素受影响，从左上角开始把原始位图覆盖到目标位图上，如果从0，0开始则传递*new Point()* 作为参数即可

操作符参数可使用6个字符之一，它们是 <, <=, >, >=, ==, 和!=。例如：如果指定为< 作为操作符，如果值小于阀值则通过，否则失败。

threshold参数即每个象素都与该阀值进行比较，可以指定为32位的数字。

mask 参数的作用是过滤颜色通道，可指定为十六进制数值，如果值为00则过滤掉相应通道，FF即表示使用相应通道。例如0x00FF0000 表示过滤掉红色通道，0xFF000000 过滤掉alpha通道 。

接下来的两个参数决定比较何时通过或失败， color 参数是设置到目标位图的象素颜色，copySource 参数决定比较失败将如何处理，如果比较结果为true，则把sourceBitmap的象素值拷贝到目标位图上，如果失败则不拷贝任何东西。

好了，现在我们看些例子：

```
var srcBmp:BitmapData = new BitmapData(stage.stageWidth,
                                       stage.stageHeight,
                                       true, 0xffffffff);
srcBmp.perlinNoise(200, 100, 2, 1000, false, true, 1, true);
var destBmp:BitmapData = new BitmapData(stage.stageWidth, stage.stageHeight, true, 0xffffffff);
var image:Bitmap = new Bitmap(destBmp);
addChild(image);
destBmp.threshold(srcBmp,                  // sourceBitmap
                  srcBmp.rect,             // sourceRectangle
                  new Point(    ),         // destPoint
                  "<",                     // operator
                  0x00880000,              // threshold
                  0x00000000,              // color
                  0x00ff0000,              // mask
                  true);                   // copySource
```

检测给定象素的红色通道（由mask定义）是否小于0x00880000。如果是，那么该象素值设为透明，如果不是，则拷贝像素。

运行上面的代码后，由布林噪波产生的暗的区域都变成了透明。

如果把操作符改为" >"则会得到相反的效果，即亮的区域都变剪切掉了。

把阀值设高或低，如0x00330000 或 0x00AA0000，则剪切更多或更少的像素

试着设置不同的颜色看看有什么效果，再试试把copySource 设置为false 看到没有拷贝像素，而是留在原始位图里。

和其他*BitmapData*方法一样，如果结合其他滤镜使用会得到更好的效果：

```
destBmp.applyFilter(destBmp, destBmp.rect,
                    new Point(    ), new DropShadowFilter(    ));
```

8.13. 在位图上应用滤镜

问题

我要在*BitmapData*上加上滤镜效果

解决办法

使用*BitmapData*类的*applyFilter( )* 方法

讨论

有两种方法应用滤镜到位图上，第一种方法是使用直接调用*BitmapData*的*applyFilter( )*方法。和其他方法一样，它需要目标位图，而且也可以把其他*BitmapData* 作为原位图：

destBmp.applyFilter(*srcBmp*, *sourceRect*, *destPoint*, *filter*);

srcBmp 参数表示目标位图
sourceRect 参数指定滤镜应用到的区域
destPoint 参数表示其实坐标
filter参数即滤镜，它是*BitmapFilter* 实例

基于原始内容和滤镜参数，每个像素值都被重新计算，然后把这些值覆盖掉原始像素数据The

比如下面的例子，在鼠标周围随即设置100个白色的像素点，然后应用模糊滤镜，每一帧重复执行，当鼠标离开白色点在鼠标周围不断生成，再移动鼠标，原来地方的白色经过模糊滤镜然后慢慢消失：

```
package {
    import flash.display.Sprite;
    import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.filters.BlurFilter;
    import flash.events.Event;
    import flash.geom.Point;
    public class FilteredBitmap extends Sprite {
        private var _bitmap:BitmapData;
        private var _image:Bitmap;
        private var _blurFilter:BlurFilter;
        public function FilteredBitmap(  ) {
            _bitmap = new BitmapData(stage.stageWidth, stage.stageHeight,
```

```
                                        false, 0xff000000);
            _image = new Bitmap(_bitmap);
            addChild(_image);
            _blurFilter = new BlurFilter(    );
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
        }
        public function onEnterFrame(event:Event):void {
            for(var i:int = 0; i < 100; i++) {
                _bitmap.setPixel(mouseX + Math.random(    ) * 20 - 10,
                                 mouseY + Math.random(    ) * 20 - 10,
                                 0xffffffff);
            }
            _bitmap.applyFilter(_bitmap, _bitmap.rect, new Point(    ), _blurFilter);
        }
    }
}
```

正如你说看到的那样，有损方法是如此有创造性，现在对比下无损方法，它使用两个位图，一个为原位图，一个为目标位图，滤镜应用在原位图上，把结果存在目标位图上，这就是无损的，因为滤镜不会改变原位图的像素值：

```
package {
    import flash.display.Sprite;
    import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.filters.BlurFilter;
    import flash.events.Event;
    import flash.geom.Point;
    public class FilteredBitmap2 extends Sprite {
        private var _bitmap:BitmapData;
        private var _bitmap2:BitmapData;
        private var _image:Bitmap;
        private var _blurFilter:BlurFilter;
        public function FilteredBitmap2(    ) {
```

```
            _bitmap = new BitmapData(stage.stageWidth, stage.stageHeight,
                                     false, 0xff000000);
            _bitmap2 = new BitmapData(stage.stageWidth, stage.stageHeight,
                                      false, 0xff000000);
            _image = new Bitmap(_bitmap);
            addChild(_image);
            _blurFilter = new BlurFilter(    );
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
        }
        public function onEnterFrame(event:Event):void {
            for(var i:int = 0; i < 100; i++) {
            _bitmap2.setPixel(mouseX + Math.random(    ) * 20 - 10,
                              mouseY + Math.random(    ) * 20 - 10,
                             0xffffffff);
            }
            _bitmap.applyFilter(_bitmap2, _bitmap.rect, new Point(    ), _blurFilter);
        }
    }
}
```

另一个应用滤镜的方法就是*Bitmap*的filters属性来控制*BitmapData*。这是另一种应用滤镜的无损方法，它不会直接接触*BitmapData*的像素数据，因为滤镜只应用在包装*BitmapData*的*Bitmap*上，看下面的代码：

```
package {
    import flash.display.Sprite;
    import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.filters.BlurFilter;
    import flash.events.Event;
    public class FilteredBitmap3 extends Sprite {
        private var _bitmap:BitmapData;
        private var _image:Bitmap;
```

```
        public function FilteredBitmap3(    ) {
            _bitmap = new BitmapData(stage.stageWidth, stage.stageHeight,
                                     false, 0xff000000);
            _image = new Bitmap(_bitmap);
            addChild(_image);
            _image.filters = [new BlurFilter(    )];
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
        }
        public function onEnterFrame(event:Event):void {
            for(var i:int = 0; i < 100; i++) {
                _bitmap.setPixel(mouseX + Math.random(    ) * 20 - 10,
                                 mouseY + Math.random(    ) * 20 - 10,
                                 0xffffffff);
            }
        }
    }
}
```

8.14. 在两幅位图之间转换

问题

我要从一幅位图中淡出到另一幅位图

解决办法

使用*BitmapData*类的*pixelDissolve()*方法

讨论

和许多*BitmapData* 方法类似，*pixelDissolve()* 使用一个原位图和目标位图，每次调用时，从原位图上拷贝指定数量随机位置的像素到目标位图上，要完成转换，须重复调用，因此可以放在*enterFrame*事件处理函数里或基于时间的函数。

第一次调用*pixelDissolve()*方法时，通过随机种子产生随机位置的像素拷贝，返回的值可以作为下一次的随机种子

该方法语法：

```
seed = srcBmp.pixelDissolve(destBmp, sourceRect, destPoint,
                            seed, numPixels, fillColor);
```

sourceRect 和 *destPoint* 参数与*BitmapData* 的其他方法一样，*seed* 在第8.10节 讲过。
numPixels 参数决定一次拷贝多少像素
fillColor 提供更简单的方法从单个位图淡出到指定颜色，如果你的原位图和目标位图一样，做转换是无效果的，这时就要用这个参数来填充颜色。

第一个例子使用两幅位图，一幅白色，一幅黑色，每帧拷贝1000个像素：

```
package {
    import flash.display.Sprite;
    import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.events.Event;
    import flash.geom.Point;
    public class Dissolve extends Sprite {
        private var _bitmap:BitmapData;
        private var _bitmap2:BitmapData;
        private var _image:Bitmap;
        private var _seed:Number;
```

```
        private var _pixelCount:int = 0;
        public function Dissolve(    ) {
            _bitmap = new BitmapData(stage.stageWidth,
                                     stage.stageHeight,
                                     false, 0xffffffff);
            _bitmap2 = new BitmapData(stage.stageWidth,
                                      stage.stageHeight,
                                      false, 0xff000000);
            _image = new Bitmap(_bitmap);
            addChild(_image);
            _seed = Math.random(    ) * 100000;
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
        }
        public function onEnterFrame(event:Event):void {
            _seed = _bitmap.pixelDissolve(_bitmap2,
                                          _bitmap.rect,
                                          new Point(    ),
                                          _seed,
                                          1000);
            _pixelCount += 1000;
            if(_pixelCount > _bitmap.width * _bitmap.height) {
                removeEventListener(Event.ENTER_FRAME,
                                    onEnterFrame);
            }
        }
    }
}
```

这里我们注意到*pixelDissolve()*方法本身不会判断是否已经转换完成，也就说像素已经拷贝完毕了，这时我们需要通过每次的像素拷贝数来计算什么时候完成转换，也就是拷贝的像素总数是否等于位图的总像素数（位图的长*位图的宽）。

下面的例子，把同样的位图作为源和目标，黑色作为填充色：

```
package {
```

```
import flash.display.Sprite;
import flash.display.Bitmap;
import flash.display.BitmapData;
import flash.events.Event;
import flash.geom.Point;
public class Dissolve2 extends Sprite {
        private var _bitmap:BitmapData;
        private var _image:Bitmap;
        private var _seed:Number;
        private var _pixelCount:int = 0;
        public function Dissolve2(    ) {
                _bitmap = new BitmapData(stage.stageWidth,
                                                stage.stageHeight,
                                                false,
                                                0xffffffff);
                _image = new Bitmap(_bitmap);
                addChild(_image);
                _seed = Math.random(    ) * 100000;
                addEventListener(Event.ENTER_FRAME, onEnterFrame);
        }
        public function onEnterFrame(event:Event):void {
                _seed = _bitmap.pixelDissolve(_bitmap,
                                                        _bitmap.rect,
                                                        new Point(    ),
                                                        _seed,
                                                        1000,
                                                        0xff000000);
                _pixelCount += 1000;
                if(_pixelCount > _bitmap.width * _bitmap.height)
                {
                        removeEventListener(Event.ENTER_FRAME, onEnterFrame);
```

```
                }
            }
        }
}
```

这种效果经常应用到两个照片之间的切换，如果想加快转换速度可以调大numPixels 参数值：

```
var numPixels:Number = _bitmap.width * _bitmap.height / 100;
_seed = _bitmap.pixelDissolve(_bitmap,
                                        _bitmap.rect,
                                        new Point(    ),
                                        _seed,
                                        numPixels,
                                        0xff000000);
_pixelCount += numPixels;
```

每次拷贝 1% 的总像素数，如果是 30 帧播放的话，也就 3 秒钟即可转换完成。

8.15. 滚动位图

问题

我想滚动显示位图

解决办法

使用*BitmapData*类的*scroll()*方法

讨论

这个方法很简单，你只要传递要滚动的具体X和Y坐标即可：

_bitmap.scroll(*xAmount*, *yAmount*);

该方法实际上是按照指定的便宜量拷贝像素。

通过在*enterFrame* 事件处理函数中或基于时间的函数中调用*scroll()*就会形成滚动的动画，下面的代码演示了滚动布林噪波产生的图案：

```
public function Scroll(    ) {
        _bitmap = new BitmapData(stage.stageWidth, stage.stageHeight,
                                        false, 0xffffffff);
        _bitmap.perlinNoise(100, 100, 3, 1000, true, true, 1, true);
        _image = new Bitmap(_bitmap);
        addChild(_image);
        addEventListener(Event.ENTER_FRAME, onEnterFrame);
}
public function onEnterFrame(event:Event):void {
        _bitmap.scroll(-1, -1);
}
```

9.0. 简介

flash.text.TextField 类是Flash播放器中专门用于显示文本信息，还有*TextArea*和*TextInput* 两个组件也和文本显示有关。*TextField* 类支持CSS和嵌入字体，这一章我们将讨论和文本有关的系统功能。

可视化类都在*flash.display*包中，*TextField*类也是，因此在使用它之前记得先导入它：

```
import flash.text.TextField;
```

ActionScript 3.0 的可视化对象列表和之前版本的ActionScript有较大不同，在之前的ActionScript中，像下面的代码那样只要调用*TextField*的构造函数就可以显示文本框了：

```
var field:TextField = new TextField(    );
```

不过在ActionScript 3.0中，创建的文本框对象不会自动加入到可视化对象列表，这就意味它不会被显示出来，只有手动调用*addChild()*方法添加它才能显示。在第六章讲过，所有的可视化容器如*Sprite*都有*addChild()*方法.下面的代码创建了*TextField* 对象并显示之：

```
package {
   import flash.display.Sprite;
   import flash.text.TextField;
   public class TextExample extends Sprite {
      public function TextExample(    ) {
         var field:TextField = new TextField(    );
         addChild(field);
      }
   }
}
```

9.1. 创建带边框的文本框

问题

我要创建带边框的文本框

解决办法

这是文本框的border属性为true。另外可通过borderColor 属性改变边框颜色

讨论

默认情况，文本框没有显示边框，这样是为了达到通用设计，例如，不想项目名称有边框出现，但是有时候我们又需要加上边框，比如说用户输入框要加上边框，这样用户就看的清除哪里是输入信息的地方了。要显示边框也很简单，只要设置border属性为true即可：

```
field.border = true;
```

若要不显示则设为false。

边框默认的颜色为黑色，若想改变颜色可设置borderColor属性，它接受十六进制的RGB数字：

```
field.borderColor = 0xFF00FF;
```


9.2. 创建带背景色的文本框

问题

我要改变文本框的背景颜色

解决办法

设置background 属性为true，另外可通过backgroundColor 属性设置背景色

讨论

默认下文本框的背景色是不显示的，可通过background 属性为true显示背景色：

```
field.background = true;
```

默认的背景色为白色，通过backgroundColor 属性可设置背景色：

```
field.backgroundColor = 0x00FFFF;
```

9.3. 变成一个用户输入框

问题

我想要让用户可以输入文本

解决办法

设置文本框的type属性为*TextFieldType.INPUT.*

讨论

文本框有两种类型： dynamic 和 input，默认为dynamic。表示可以由ActionScript控制，但是用户不能输入数据，设置成INPUT就可以了，该值是*flash.display.TextFieldType* 类常量：

```
field.type = TextFieldType.INPUT;
```

再加上边框和背景色

```
field.border = true;
field.background = true;
```

另外文本框的selectable 属性必须为true，否则虽可输入的但是无法决定输入位置。

dynamic表示是动态文本框，且不允许输入数据，DYNAMIC是 *flash.display.TextFieldType* 类常量：

```
field.type = TextFieldType.DYNAMIC;
```


9.4. 变成一个密码输入框

问题

我要如何创建密码输入框呢，能隐藏输入的字符信息，而且不允许复制

解决办法

设置文本框的displayAsPassword属性为true.

讨论

在密码框输入信息是看不到具体内容的，这是最基本的保密手段，要创建类似功能的密码框，只要把*TextField*的displayAsPassword属性设为TRue即可：

```
field.displayAsPassword = true;
```

这样所有输入的内容都变成了*：

```
field.displayAsPassword = true;
field.text = "example text";   // 显示: **********
```

密码框同时也屏蔽了复制功能，也是一种必要的安全措施，反之别人盗走密码内容。

9.5. 过滤文本输入框

问题

我想过滤掉用户输入的某些内容

解决办法

设置文本框的restrict 属性

讨论

默认下用户可输入任何字符，但是有些情况我们希望只能输入合法的字符，比如电话号码文本框只能输入数字和横线。

TextField 的restrict 属性可以指定允许的字符被输入，比如：下面的字符是允许的：

```
field.restrict = "abcdefg";
```

这样就只能输入a, b, c, d, e, f, 或g了，其他字符都被过滤掉了。
如果restrict设为空字符串，那么代表可以输入任何字符，如果想过滤所有字符，则可把type属性设为DYNAMIC。

另外我们还要注意字符是有大小写之分的，换句话说字符a和A是不同的，如上面的restrict 属性设置为*abcdefg*，但是AB等大写字符仍然可以输入。

restrict 属性支持一定的正则表达式，如果你想指定一个范围的字符可用-符号分开，像下面那样：

```
field.restrict = "a-zA-Z";     // 只允许大小写字母
field.restrict = "a-zA-Z ";    // 只允许大小写字母和空格
field.restrict = "0-9";        // 只允许数字
```

另外也可指定不允许的字符输入，不过需要（^）开头：
下面的例子都是设置为不允许的：

```
field.restrict = "^abcdefg"; //不允许a 到 g 的字母
field.restrict = "^a-z";       // 不允许所有小写字母
field.restrict = "0-9^5";      // 允许除了5之外的数字
```

还可以指定Unicode 编码的字符，比如不允许(Control-Z) 产生的字符：

```
field.restrict = "^\u001A";
```

还有些特殊符号，可通过（\\）进行转义：

```
field.restrict = "0-9\\-";        // 允许数字和横线
field.restrict = "0-9\\^";        //允许数字和^
field.restrict = "0-9\\\\";       // 允许数字和斜杆
```

restrict 属性只影响用户输入的字符进行过滤，但不会对程序产生的字符串进行过滤。

9.6. 设置输入域的最大数量

问题

我要限制输入字符的个数

解决办法

设置maxChars属性

讨论

默认下输入框可以输入任意多的内容，但是最好是控制输入的个数，比如有个文本框输入用户2个字符的国家代码，没有必要输入这么多，只需要2个字符即可，这时我们可以通过maxChars 属性进行限制：

```
field.maxChars = 6;   // 最大6个字符
```

如果设为 null 则表示不限制。

9.7. 显示文本

问题

如何设置显示的文本呢

解决办法

设置text 属性

讨论

在用户输入之前可以在文本框里预先设置些提示信息，这时可通过text 属性进行设置：

```
field.text = "this will display in the field";
```

包括些特殊字符\\t 表示制表符，\\n 表示换新行都可以用。

还可以通过*appendText()*方法追加字符串：

```
field.appendText("new text");
```

9.8. 显示HTML格式文本

问题

我想在文本框中显示HTML格式内容

解决办法

设置htmlText 属性值为HTML内容

讨论

文本框支持基本的HTML标签：

```
field.htmlText = "<u>显示带下划线的文本.</u>";
```

text 属性只渲染为不同的文本信息，这就意味着即使在text 属性中设置为 <u>test</u>，但显示出来的还是<u>test</u>。如果你想显示出Html代码则可以显示在text 属性里：

```
field.text = "<u>underlined text</u>";
/* 显示为：
<u>underlined text</u>
*/
```

利用这一点，我们可以既显示HTML内容又可显示HTML代码：

```
htmlCode = "<i>italicized text</i>";
sourceHTML.text = htmlCode;
renderedHTML.htmlText = htmlCode;
```

目前文本框支持的 HTML 标签有：<b>, <i>, <u>, <font> (包括 face, size, 和 color 属性), <p>,
, <a >, <li>, <img>, 和<textformat> (包括 leftmargin, rightmargin, blockindent, indent, leading, 和 tabstops 属性).

9.9. 压缩空格

问题

我想显示HTML内容时压缩空格

解决办法

设置condenseWhite 属性为true

讨论

当在文本框里显示HTML内容时，可通过condenseWhite属性压缩空格，大多数浏览器都这么做。例如，下面的文本在浏览器里渲染时只包含一个空格，而原始是多个空格，这是被压缩的结果。

hello friend

而在ActionScript的文本框显示HTML内容时也可进行压缩空格，只要把condenseWhite 属性设为true：

```
field.condenseWhite = true;
field.htmlText = "hello friend";    // 显示: "hello friend"
```

condenseWhite 属性只当 htmlText 有内容时有效。

9.10. 调整文本框大小以适应内容

问题

我要调整文本框大小使之正好容纳下内容即可

解决办法

使用autoSize属性

讨论

设置autoSize属性可自动根据内容调整文本框大小。可用值为RIGHT, LEFT, CENTER, 和NONE，都是 *flash.text.TextFieldAutoSize* 类常量。默认值为NONE，表示不自动调整大小。

当设为LEFT 时表示大小变化时左上角位置保持不动，也就是说右下角的位置根据内容变化动态调整：

```
// 下面的两句代码效果一样
field.autoSize = TextFieldAutoSize.LEFT;
field.autoSize = true;
```

设置为CENTER时根据文本框的中心，保持顶部位置不变，调整左右边框：

```
field.autoSize = TextFieldAutoSize.CENTER;
```

设置为RIGHT 时保持右上角位置不变，根据内容变化自动调整左下角位置：

```
field.autoSize = TextFieldAutoSize.RIGHT;
```

*wordWrap*属性设为true时当内容超出范围时自动移到下一行，也就说保持水平宽度不变，如果为false则自动调整水平宽度不换行：

```
var field:TextField = new TextField(    );
field.autoSize = TextFieldAutoSize.LEFT;
field.text = "Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi tortor purus, aliquet a, ornare ac, suscipit a, est. Nullam hendrerit molestie erat. Nunc nulla tortor, ullamcorper et, elementum vel, fringilla sed, dui. Praesent fermentum interdum orci.";
addChild(field);
```

下面再添加一行字符串：

```
var field:TextField = new TextField(    );
field.autoSize = TextFieldAutoSize.LEFT;
field.text = "Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi tortor purus, aliquet a, ornare ac, suscipit a, est.";
field.text += "\n";
field.text += "Nullam hendrerit molestie erat. Nunc nulla tortor, ullamcorper et, elementum vel, fringilla sed, dui. Praesent fermentum interdum orci.";
addChild(field);
```

wordWrap 属性为true进行自动换行：

```
var field:TextField = new TextField(    );
field.autoSize = TextFieldAutoSize.LEFT;
field.wordWrap = true;
field.text = "Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi tortor purus, aliquet a, ornare ac, suscipit a, est. Nullam hendrerit molestie erat. Nunc nulla tortor, ullamcorper et, elementum vel, fringilla sed, dui. Praesent fermentum interdum orci.";
addChild(field);
```

9.11. 滚动文本

问题

我要用ActionScript控制文本滚动

解决办法

使用scrollV, maxScrollV, bottomScrollV, scrollH, 和maxScrollH 属性和设置鼠标滚轮相关的WheelEnabled 属性

讨论

在没有滚动条的情况下我们仍然可以通过ActionScript控制文本框的文本滚动，我们可以通过scrollV, maxScrollV控制垂直滚动或scrollH，maxScrollH控制水平滚动。

文本框有个行数，从1开始，当行数过多时有些行数的内容可能不能显示，这时可通过滚动来显示出来。下面先来了解下三个属性：scrollV, bottomScrollV, 和maxScrollV。
scrollV 属性为可读写属性，表示文本框能显示出来的最顶行行数，要想滚动文本框，只要设置scrollV属性值。例如下面的代码：

```
field.scrollV = 1;     //滚动到最顶行
field.scrollV += 1;    // 向下滚动一行
field.scrollV = 6;     // 滚动6行
```

如果想滚动到下一页则需设置bottomScrollV属性，它表示显示的最后一行函数。比如bottomScrollV值为13，如果scrollV设置为6，则bottomScrollV 自动调整为18：

```
// 滚动一页
field.scrollV = field.bottomScrollV;
// 完全滚动一页，包括最后一行
field.scrollV = field.bottomScrollV + 1;
```

通过maxScrollV属性可滚动到最后一页，该属性是只读的，表示文本的最大行数。

因此在设置scrollV时其值不能小于1也不能超过maxScrollV。如果超过这个范围则赋值无效。

```
field.scrollV = field.maxScrollV;      // 滚动到最后页
```

所有的垂直滚动属性都是以行为单位，但是水平滚动属性(scrollH和maxScrollH)则是以像素为单。scrollH表示每次水平滚动的像素值，maxScrollH 表示滚动到最右边的最大值：

```
field.scrollH = 0;                     // 滚动到最左边
field.scrollH += 1;                    // 向右滚动1像素
field.scrollH = field.maxScrollH;      // 滚动到最右边
```

另外*mouseWheelEnabled* 属性允许鼠标滚轮来控制文本框滚动，默认为false：

```
field.mouseWheelEnabled = true;
```

9.12. 响应滚动事件

问题

我要监控文本框的滚动

解决办法

监听滚动事件

讨论

当水平或垂直滚动产生时会发出scroll事件，*flash.events.Event* 类的SCROLL 常量即代表该事件，下面的代码注册文本框的scroll事件监听器：

```
field.addEventListener(Event.SCROLL, onTextScroll);
```

*onTextScroll()*方法处理滚动事件：

```
private function onTextScroll(event:Event):void {
   trace("scroll");
}
```


9.13. 格式化文本

问题

我要格式化文本框文本

解决办法

使用HTML标签，或传递*TextFormat*对象给*TextField.setTextFormat()*方法，或设置文本框的styleSheet属性

讨论

应用格式化文本有以下几种方式：

使用HTML标签进行格式化，如 <font>, <b>, 和 <u> 标签。
使用*TextFormat*对象。
使用CSS。

三种方式各有优缺点，HTML格式化可能是最简单的，但是难于管理，使用*TextFormat*对象比HTML复杂，如果要应用复杂的格式化，它比HTML格式化要好用的多。通过*StyleSheet*对象使用CSS样式是最具灵活性的，它可以读取CSS文档应用到SWF文件上而不需要每次都重新编译。

最快最简单的方式就是使用HTML标签。例如，下面的代码显示粗体和下划线文字：

```
field.html = true;
field.htmlText = "<b>Bold text</b> <u>Underlined text</u>";
```

使用*TextFormat*对象进行格式化稍微复杂些，首先要先创建*TextFormat*对象：

```
var formatter:TextFormat = new TextFormat(    );
```

接着，对*TextFormat*对象的一些属性进行赋值：

```
formatter.bold = true;            // 设置粗体
formatter.color = 0xFFFF00;    // 设置文本颜色为黄色
formatter.blockIndent = 5;     //调整间距为5
```

通过*setTextFormat()*方法应用：

```
field.setTextFormat(formatter);
```

通过*setTextFormat()*方法格式化只对当前的文本起作用，如果之后又加入文本则不会进行格式化：

```
field.text = "this is sample text";
field.setTextFormat(formatter);       // 应用格式化
field.text = "this is new text";     // 没有应用格式化
field.setTextFormat(formatter);       // 再次应用格式化
field.text += "appended text";        // 格式化被取消
```

如果改变了*TextFormat*对象则需重新调用*setTextFormat()*方法。

使用*flash.text.StyleSheet*类可支持CSS。*StyleSheet*构造函数不需要任何参数：

```
var css:StyleSheet = new StyleSheet(    );
```

下面的表格列出了 CSS 属性和相应的 ActionScript 属性：

CSS 属性	ActionScript 属性	描述
color	color	十六进制值 #RRGGBB
display	display	显示文本方式：inline,block,none
font-family	fontFamily	字体类型
font-size	fontSize	字体大小
font-style	fontStyle	normal 或 italic
font-weight	fontWeight	normal 或 bold
kerning	kering	true 或 false,只对嵌入字体有效
margin-left	marginLeft	左边距
margin-right	marginRight	右边距
text-align	textAlign	对齐方式：left,center,right,justify
text-decoration	textDecoration	none,underline

text-indent	textIndent	缩进

通过构造函数生成一个*StyleSheet* 对象，然后传递给*setStyle()*方法。一个样式表对象实际上是一个ActionScript样式数组，看下面的例子：

```
var sampleStyle:Object = new Object(    );
sampleStyle.color = "#FFFFFF";
sampleStyle.textAlign = "center";
```

下面的代码等同于上面的代码：

```
var sampleStyle:Object = {color: "#FFFFFF", textAlign: "center"};
```

然后调用类的*setStyle()*方法应用样式：

```
css.setStyle(".sample", sampleStyle);
```

虽然我们可以这样定义，但实际应用中很少这样，一般都是读取CSS文档，这样改变样式时只要修改CSS文档也不必每次重新编译.swf文件。

可通过*flash.net.URLLoader*类载入CSS文档。一旦读入文档，只要生成一个*StyleSheet*对象然后传递给对象的*parseCSS()*方法即可。下面的例子载入*styles.css*，一个CSS文档：

```
p {
   font-family: _sans;
   color: #FFFFFF;
}
.emphasis {
   font-weight: bold;
   font-style: italic;
}
```

```
package {
   import flash.display.Sprite;
   import flash.text.TextField;
   import flash.events.Event;
   import flash.text.TextFieldAutoSize;
   import flash.text.StyleSheet;
   import flash.net.URLLoader;
   import flash.net.URLRequest;
   public class CSSText extends Sprite {
```

```
        public function CSSText(    ) {
                var loader:URLLoader = new URLLoader(    );
                loader.addEventListener(Event.COMPLETE, onLoadCSS);
                var request:URLRequest = new URLRequest("styles.css");
                loader.load(request);
        }
        private function onLoadCSS(event:Event):void {
           var css:StyleSheet = new StyleSheet(    );
           css.parseCSS(URLLoader(event.target).data);
                var field:TextField = new TextField(    );
                field.autoSize = TextFieldAutoSize.LEFT;
                field.wordWrap = true;
                field.width = 200;
                addChild(field);
                field.styleSheet = css;
                field.htmlText = "<p><span class='emphasis'>Lorem ipsum</span> dolor sit amet, consectetuer adipiscing elit. Morbi tortor purus, aliquet a, ornare ac, suscipit a, est. Nullam hendrerit molestie erat. Nunc nulla tortor, ullamcorper et, elementum vel, fringilla sed, dui. Praesent fermentum interdum orci.</p>";
        }
    }
}
```


这里需要注意几点：

只有当文本框渲染为HTML时才可应用CSS。

HTML和CSS必须配合，比如CSS里定义了一个叫*someCSSClass*的类，那么HTML也必须有。

必须先应用CSS，再应用HTML。

如果用户想选择不同的CSS样式，我们可把HTML文本存在变量或数组中，这样每次新的CSS被载入时就可以重新应用了：

```
package {
   import flash.display.Sprite;
   import flash.text.TextField;
   import flash.events.Event;
   import flash.events.MouseEvent;
```

```
import flash.text.TextFieldAutoSize;
import flash.text.StyleSheet;
import flash.net.URLLoader;
import flash.net.URLRequest;
public class CSSText extends Sprite {
    private var _field:TextField;
    private var _html:String;
    public function CSSText(    ) {
        var css1:TextField = new TextField(    );
        css1.text = "stylesheet 1";
        css1.selectable = false;
        var css1Container:Sprite = new Sprite(    );
        css1Container.addEventListener(MouseEvent.CLICK, onCSS1);
        css1Container.addChild(css1);
        addChild(css1Container);
        var css2:TextField = new TextField(    );
        css2.text = "stylesheet 2";
        css2.selectable = false;
        var css2Container:Sprite = new Sprite(    );
        css2Container.addEventListener(MouseEvent.CLICK, onCSS2);
        css2Container.addChild(css2);
        addChild(css2Container);
        css2Container.y = 25;
            _field = new TextField(    );
            _field.autoSize = TextFieldAutoSize.LEFT;
            _field.wordWrap = true;
            _field.width = 200;
            addChild(_field);
            _html = "<p><span class='emphasis'>Lorem ipsum</span> dolor sit amet, consectetuer adipiscing elit. Morbi tortor purus, aliquet a, ornare ac, suscipit a, est. Nullam hendrerit molestie erat. Nunc nulla tortor, ullamcorper et, elementum vel, fringilla sed, dui. Praesent fermentum interdum orci.</p>";
```

```
            _field.y = 50;
        }
        private function loadCSS(url:String):void {
                var loader:URLLoader = new URLLoader(    );
                loader.addEventListener(Event.COMPLETE, onLoadCSS);
                var request:URLRequest = new URLRequest(url);
                loader.load(request);
        }
        private function onCSS1(event:MouseEvent):void {
            loadCSS("styles.css");
        }
        private function onCSS2(event:MouseEvent):void {
            loadCSS("styles2.css");
        }
        private function onLoadCSS(event:Event):void {
            var css:StyleSheet = new StyleSheet(    );
            css.parseCSS(URLLoader(event.target).data);
                _field.styleSheet = css;
                _field.htmlText = _html;
        }
    }
}
```


9.14. 格式化用户输入的文本

问题

我要对用户输入的文本进行格式化

解决办法

应用*TextFormat* 对象到文本框的*defaultTextFormat* 属性上

讨论

通过使用*defaultTextFormat* 属性即可对用户输入的文本进行格式化，只要创建一个 *TextFormat* 对象赋值给*defaultTextFormat* 属性：

```
var formatter:TextFormat = new TextFormat(    );
formatter.color = 0x0000FF;          // 设置颜色为蓝色
field.defaultTextFormat = formatter;
```

9.15. 格式化一部分文本

问题

我想只格式化一部分文本而不是全部，或者应用多种格式化到文本的不同部分上

解决办法

创建*TextFormat* 对象的*setTextFormat()* 方法对部分文本进行格式化

讨论

第9.13节已经讨论了格式化整个文本，实际上通过*setTextFormat()*方法可以格式化部分文本，需要增加其他参数：

index

对对应位置的字符格式化。

textFormat

*TextFormat*对象引用。

下面的例子对第一个字符格式化：

```
field.setTextFormat(0, formatter);
```

要想对一定范围的字符进行格式化，需要给出三个参数：

startIndex

格式化起始位置

endIndex

格式化结束位置

textFormat

TextFormat 对象引用

下面的例子对开始的前10个字符格式化：

```
field.setTextFormat(0, 10, formatter);
```

9.16. 设置文本框字体

问题

我要更改文本框的字体

解决办法

使用HTML的<font>标签，或者设置*TextFormat*对象的font属性，或者通过CSS的font-family属性

讨论

修改字体有多种方法，如果使用HTML的话可通过<font> 标签更改：

```
field.htmlText = "<font face='Arial'>Formatted text</font>";
```

也可设置*TextFormat*对象的font属性：

```
formatter.font = "Arial";
```

或者在CSS中定义font-family 属性：

```
p {
   font-family: Arial;
}
```

需要注意的是电脑中必须要有你所指定的字体，因为有些电脑上可能没有安装相应的字体，这是可指定多种字体：

```
formatter.font = "Arial, Verdana, Helvetica";
```

如果都没有指定字体，默认使用系统字体。另外我们还可使用字体组，字体组是系统默认字体的一个分类，有三种： *_sans*, *_serif*, 和*_typewriter*。 *_sans* 组包含如 Arial 或 Helvetica，*_serif* 组包含如 Times 或 Times New Roman，*_typewriter* 组包含如 Courier 或 Courier New。

9.17. 嵌入字体

问题

我要嵌入自己的字体

解决办法

通过[Embed] 元数据嵌入字体，设置文本框的embedFonts 属性为true，通过<font> 标签，*TextFormat* 对象或CSS应用字体

讨论

当用户电脑上没有相应字体时可以通过[Embed]元数据把字体嵌入到swf中，[Embed] 元数据在类外面申明，可以嵌入TrueType 字体或系统字体，语法如下：

```
[Embed(source="pathToTtfFile", fontName="FontName", mimeType="application/x-font-truetype")]
```

TrueType f字体路径可以是相对或绝对路径，比如：

```
[Embed(source="C:\Windows\Fonts\Example.ttf",fontName="ExampleFont",
mimeType="application/x-font-truetype")]
```

fontName 属性可被CSS或ActionScript引用。

嵌入系统字体语法类似，只是*systemFont* 属性代替了source 属性。*systemFont* 属性指定嵌入的系统字体，下面的例子嵌入Times New Roman字体：

```
[Embed(systemFont="Times New Roman",fontName="Times New Roman",
mimeType="application/x-font-truetype")]
```


嵌入好字体后，下一步就是告诉文本框使用嵌入的字体，可设置embedFonts 属性为true：

```
field.embedFonts = true;
```

接下来通过<font> 标签，*TextFormat* 对象或CSS使用之。例如，如果*fontName* 值为Times New Roman：

```
formatter.font = "Times New Roman";
```

使用<font>标签：

```
field.htmlText = "<font family='Times New Roman'>Example</font>";
```

使用CSS：

```
var css:StyleSheet = new StyleSheet(    );
css.setStyle("p", {fontFamily: "Times New Roman"});
field.htmlText = "<p>Example</p>";
```

9.18. 创建可以被旋转的文字

问题

我要使一些文字在旋转时仍能正确显示

解决办法

使用嵌入字体

讨论

一般情况下默认字体是最好的，但是有些特殊情况如文本框被旋转或者它的父容器被旋转，这时默认字体渲染得文字就不能显示了，这时候必须使用嵌入字体。

9.19. 显示Unicode编码的文字

问题

我要显示Unicode编码的文字，可能包括非英文字符

解决办法

载入外部源文本，使用Unicode转义序列将字符赋值给文本框的text 属性

讨论

如果想在文本框中显示Unicode文本，可通过以下几种方式：

载入外部Unicode源数据(如文本文件，XML文档，数据库)。
直接使用字符。
使用Unicode转义序列。

在支持Unicode的编辑器（如Flex Builder），可直接输入字符，因为都是经过Unicode编码过了，如果你知道字符的转义序列，也可把它赋值给文本框的text属性，在ActionScript里Unicode转义序列以\\u开头后面跟着4个十六进制数字，而且要放在括号中，如下面的例子：

```
field.text = "Add a registered mark directly (®) or with a Unicode escape sequence (\u00AE)";
```

9.20. 设置文本框的焦点

问题

我想用ActionScript 设置文本框焦点

解决办法

使用*Stage.focus* 属性

讨论

使用*Stage.focus*属性可在程序里赋值焦点到一个文本框上，每个可视化对象都有一个stage 属性，它是*Stage*类实例，下面的代码把焦点赋值给叫field的文本框上：

```
stage.focus = field;
```

当一个*.swf* 第一次被载入到浏览器时，它是没有焦点的，因此在必须先设定把焦点移到Flash播放器上，下面的例子把焦点赋值给文本框：

```
package {
   import flash.display.Sprite;
   import flash.text.TextField;
   import flash.text.TextFieldType;
   import flash.events.MouseEvent;
   public class TextExample extends Sprite {
      public function TextExample( ) {
         var field:TextField = new TextField( );
         field.border = true;
         field.background = true;
         field.type = TextFieldType.INPUT;
         addChild(field);
         var button:Sprite = new Sprite( );
         button.graphics.lineStyle( );
         button.graphics.beginFill(0xFFFFFF);
         button.graphics.drawRect(0, 0, 100, 50);
         button.graphics.endFill( );
         button.addEventListener(MouseEvent.CLICK, onClick);
         button.y = 100;
         addChild(button);
```

```
        }
        private function onClick(event:MouseEvent):void {
            stage.focus = TextField(getChildAt(0));
        }

    }
}
```

把*Stage.focus* 设为null即可移除焦点：

```
stage.focus = null;
```

9.21. 用ActionScript实现选择文本

问题

我要选中一部分文字

解决办法

使用*TextField.setSelection()*方法。

讨论

*TextField.setSelection()*方法是以程序的方式选中一部分文本，它接受两个参数：

startIndex

开始位置

endIndex

结束位置

调用该方法之前，文本框必须先拥有焦点，可通过*Stage.focus*进行设置：

```
stage.focus = field;                            //设置焦点
field.text = "this is example text";    //  设置文本内容
field.setSelection(0, 4);                     //选中"this"
```

通过 selectionBeginIndex 和 selectionEndIndex 两个只读属性可获得选中文本的具体位置。

9.22. 设置文本框的光标位置

问题

我要设置文本框的光标位置，这样可在任意位置插入文字

解决办法

使用*TextField.setSelection()*方法

讨论

仍然可通过*TextField.setSelection()* 设置文本框的光标位置，只要指定两个参数为同一个值即可，看下面的例子，记得首先要让文本框获得焦点：

```
// 设置光标为起始位置
field.setSelection(0, 0);
```

通过*caretIndex*只读属性可获得当前光标的所在位置：

```
trace(field.caretIndex);
```


9.23. 响应文本选中和取消选中事件

问题

我想当选中文本时激活某个事件执行一些任务

解决办法

监听*focusIn* 和 *focusOut* 事件

讨论

当文本框获得焦点时会激活*focusIn* 事件，失去焦点时激活*focusOut* 事件，两个事件都是*flash.events.FocusEvent* 对象。*FocusEvent* 类定义了一个叫*relatedObject*属性，当激活*focusIn* 事件时*relatedObject*属性指向获得焦点的对象，当激活*focusOut*事件时*relatedObject* 属性指向刚刚获得焦点的对象。通过*flash.events.FocusEvent* 类的FOCUS_IN和FOCUS_OUT 常量注册事件：

```
field.addEventListener(FocusEvent.FOCUS_IN, onFocus);
```

当焦点改变时*focusIn*和*focusOut* 事件会同时激活，它们属于不可取消的事件，如果想取消这些事件则必须在触发它们之间注册监听事件。*keyFocusChange* 和 *mouseFocusChange* 是可取消的事件，可使用*FocusEvent* 类的KEY_FOCUS_CHANGE 和 MOUSE_FOCUS_CHANGE常量注册事件，*FocusEvent.preventDefault()* 方法可取消默认事件，下面的例子演示通过Tab键移动焦点，当field1没有内容时取消焦点移动：

```
package {
```

```
    import flash.display.Sprite;
    import flash.text.TextField;
    import flash.text.TextFieldType;
    import flash.events.FocusEvent;
    public class Text extends Sprite {
        private var _field1:TextField;
        private var _field2:TextField;
        public function Text(    ) {
            _field1 = new TextField(    );
            _field1.border = true;
            _field1.background = true;
            _field1.type = TextFieldType.INPUT;
            addChild(_field1);
            _field1.addEventListener(FocusEvent.KEY_FOCUS_CHANGE, onKeyFocus);
            _field2 = new TextField(    );
            _field2.border = true;
            _field2.background = true;
            _field2.type = TextFieldType.INPUT;
            addChild(_field2);
            _field2.y = 100;
        }
        private function onKeyFocus(event:FocusEvent):void {
            if(_field1.text == "") {
                    event.preventDefault(    );
            }
        }

    }
}
```

9.24. 监听用户输入的内容

问题

我想监听用户修改文本框的内容

解决办法

监听*textInput* 事件.

讨论

我们可以通过*TextEvent*事件控制用户对文本框内容的修改，如删除，剪切，插入或者拷贝等操作，对文本框的每一次修改都会激活*textInput* 事件，可通过*flash.events.TextEvent.TEXT_INPUT* 常量监听该事件：

```
field.addEventListener(TextEvent.TEXT_INPUT, onTextInput);
```

textInput 事件是一个可取消的*TextEvent* 对象，*TextEvent* 类定义了一个text 属性，它包含用户输入文本框的内容，下面的例子判断用户输入的第一个字符是否是"a"：

```
package {
   import flash.display.Sprite;
   import flash.text.TextField;
   import flash.text.TextFieldType;
   import flash.events.TextEvent;
   import flash.events.TextEvent;
   public class Text extends Sprite {
      private var _field:TextField;
      public function Text(    ) {
         _field = new TextField(    );
         _field.border = true;
         _field.background = true;
         _field.type = TextFieldType.INPUT;
         addChild(_field);
         _field.addEventListener(TextEvent.TEXT_INPUT, onTextInput);
      }
      private function onTextInput(event:TextEvent):void {
         if(event.text == "a" && _field.length == 0) {
               event.preventDefault(    );
```

```
        }
    }
  }
}
```

9.25. 在文本框里添加超链接

问题

我想在文本框里加入超链接

解决办法

利用HTML <a href> 标签设置文本框的htmlText 属性，也可用*TextFormat* 对象的url属性

讨论

两种方法都有个前提那就是文本框的html 属性必须先设为TRue：

```
field.html = true;
```

在文本框的htmlText 属性里设置HTML超链接标签<a href>：

```
field.htmlText = "<a href='http://www.rightactionscript.com'>Website</a>";
```

还可指定目标窗口的打开方式：

```
field.htmlText = "<a href='http://www.rightactionscript.com'
target='blank'>Website</a>";
```

当鼠标移到超链接上时鼠标图标会变成手型图标，下面给超链接加上下划线和颜色：

```
var htmlLink:String = "<font color='#0000FF'><u>";
htmlLink += "<a href='http://www.rightactionscript.com'>Website</a>";
htmlLink += "</u></font>";
field.htmlText = htmlLink;
```

另外用*TextFormat*对象的url属性也可达到HTML标签同样的效果：

```
field.text = "Website";
var formatter:TextFormat = new TextFormat(    );
formatter.url = "http://www.rightactionscript.com/";
field.setTextFormat(formatter);
```

通过*TextFormat* 对象的target 属性设置链接打开窗口方式：

```
field.text = "Website";
var formatter:TextFormat = new TextFormat(    );
formatter.url = "http://www.rightactionscript.com/";
formatter.target = "_blank";
field.setTextFormat(formatter);
```

加上颜色和下划线：

```
field.text = "Website";
var formatter:TextFormat = new TextFormat(    );
formatter.color = 0x0000FF;
formatter.underline = true;
formatter.url = "http://www.rightactionscript.com/";
field.setTextFormat(formatter);
```

超链接可以是http或https协议，也可以是其他协议如邮件消息：

```
field.text = "email";
var formatter:TextFormat = new TextFormat(    );
formatter.color = 0x0000FF;
formatter.underline = true;
formatter.url = "mailto:joey@person13.com";
field.setTextFormat(formatter);
```

用CSS是最完美的，它提供了超链接的a:link, a:active,和a:hover 样式：

```
var css:StyleSheet = new StyleSheet(    );
css.parseCSS("a {color: #0000FF;} a:hover {text-decoration: underline;}");
field.styleSheet = css;
field.html = true;
field.htmlText = "<a href='http://www.rightactionscript.com'>Website</a>";
```

9.26. 用超链接调用ActionScript代码

问题

我想让用户点击超链接时调用ActionScript方法

解决办法

监听*TextEvent.LINK* 事件

讨论

在ActionScript3里点击超链接调用ActionScript代码是很简单的。首先我们定义一个超链接：

```
field.htmlText = "<a href='event:http://www.rightactionscript.com'>Website</a>";
```

使用*flash.events.TextEvent.LINK* 常量注册监听器：

```
field.addEventListener(TextEvent.LINK, onClickHyperlink);
```

该 *event* 是 *flash.events.TextEvent* 实例，*event* 对象有个 text 属性包含 HRef 属性的值，也就是说上面例子中 event 的 text 属性值为 http://www.rightactionscript.com。

9.27. 高级文本布局

问题

我想要更灵活的布局方式

解决办法

使用 *numLines* 属性和*getCharBoundaries( )*, *getCharIndexAtPoint( )*, *getFirstCharInParagraph( )*, *getLineIndexAtPoint( )*, *getLineIndexOfChar( )*, *getLine- Length( )*, *getLineMetrics( )*, *getLineOffset( )*, *getLineText( )*, 和 *getParagraphLength( )* 方法

讨论

在Flash播放器8以及之前的版本很难精确控制文本布局，从8.5开始*TextField* 类定义了一系列API用于精确控制文本布局。

*TextField*类提供了两个方法获取文本信息，*getCharBoundaries( )*方法返回*flash.geom.Rectangle* 对象，它定义指定位置的字符的边界，*getCharIndexAtPoint( )*方法指定坐标的字符在文本中的位置。下面的例子使用*getCharIndexAtPoint( )* 和 *getCharBoundaries( )* 高亮显示用户选中的字符：

```
package {
   import flash.display.Sprite;
```

```
import flash.text.TextField;
import flash.events.MouseEvent;
import flash.geom.Rectangle;
public class Text extends Sprite {
    private var _field:TextField;
    private var _highlight:Sprite;
    public function Text(    ) {
        _field = new TextField(    );
        _field.border = true;
        _field.background = true;
        _field.multiline = true;
        _field.wordWrap = true;
        _field.selectable = false;
        _field.width = 400;
        _field.height = 400;
        addChild(_field);
        _field.text = "Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi tortor purus, aliquet a, ornare ac, suscipit a, est. Nullam hendrerit molestie erat. Nunc nulla tortor, ullamcorper et, elementum vel, fringilla sed, dui. Praesent fermentum interdum orci.";
        _field.addEventListener(MouseEvent.CLICK, onClick);
        _highlight = new Sprite(    );
        addChild(_highlight);
    }
    private function onClick(event:MouseEvent):void {
        var index:int = _field.getCharIndexAtPoint(mouseX, mouseY);
        var rectangle:Rectangle = _field.getCharBoundaries(index);
        _highlight.graphics.clear(    );
        _highlight.graphics.lineStyle(0, 0, 0);
        _highlight.graphics.beginFill(0x00FFFF, .25);
        _highlight.graphics.drawRect(rectangle.x, rectangle.y, rectangle.width, rectangle.height);
        _highlight.graphics.endFill(    );
    }
```

```
    }
}
```

*TextField*类也定义了一个属性和方法获取多行文本的信息。*numLines* 属性表示文本框所包含的文本行数，*getLineIndexAtPoint()* 方法返回当前坐标所在的行数，*getLineIndexOfChar()*返回指定字符所在的行数，*getLineLength()*方法返回指定行所包含的字符数，*getLineText()*方法返回指定行所包含的文本，*getLineOffset()*方法返回指定行第一个字符的位置，*getLineMetrics()*方法关于指定行数的*flash.text.TextLineMetrics*对象。*TextLineMetrics*类定义了该行文本的上位，下位，高度，宽度等等信息。

还有两个方法用于获取段落信息：*getFirstCharInParagraph()*方法返回指定段落的第一个字符所在位置，*getParagraphLength()*方法返回指定段落所包含的字符数。

下面的例子演示了上面讨论的大多数方法：

```
package {
    import flash.display.Sprite;
    import flash.text.TextField;
    import flash.events.MouseEvent;
    import flash.geom.Rectangle;
    import flash.text.TextLineMetrics;
    public class Text extends Sprite {
        private var _field:TextField;
        private var _highlight:Sprite;
        public function Text(    ) {
            _field = new TextField(    );
            _field.border = true;
            _field.background = true;
            _field.multiline = true;
            _field.wordWrap = true;
            _field.selectable = false;
            _field.width = 400;
            _field.height = 400;
            addChild(_field);
            _field.text = "Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi tortor purus, aliquet a, ornare ac, suscipit a, est. Nullam hendrerit molestie erat. Nunc nulla tortor, ullamcorper et, elementum vel, fringilla sed, dui. Praesent fermentum interdum orci.";
```

```
            _field.addEventListener(MouseEvent.CLICK, onDoubleClick);
            _highlight = new Sprite(    );
            addChild(_highlight);
        }
        private function onDoubleClick(event:MouseEvent):void {
            var index:int = _field.getCharIndexAtPoint(mouseX, mouseY);
            var startIndex:int = _field.getFirstCharInParagraph(index);
            var stopIndex:int = startIndex + _field.getParagraphLength(index);
            var startLine:int = _field.getLineIndexOfChar(startIndex);
            var stopLine:int = _field.getLineIndexOfChar(stopIndex - 1);
            var metrics:TextLineMetrics;
            var lineCharacter:int;
            var rectangle:Rectangle;
            _highlight.graphics.clear(    );
            _highlight.graphics.lineStyle(0, 0, 0);
            for(var i:int = startLine; i <= stopLine; i++) {
                lineCharacter = _field.getLineOffset(i);
                rectangle = _field.getCharBoundaries(lineCharacter);
                metrics = _field.getLineMetrics(i);
              _highlight.graphics.beginFill(0x00FFFF, .25);
              _highlight.graphics.drawRect(rectangle.x, rectangle.y, metrics.width, metrics.height);
              _highlight.graphics.endFill(    );
            }
        }

    }
}
```

9.28. 高级抗锯齿

问题

我要控制文字的抗锯齿

解决办法

对于嵌入字体可通过设置文本框的*antiAliasType*属性为*flash.text.AntiAliasType.ADVANCED*，然后设置*gridTypeFit* 和sharpness 属性

讨论

默认下文本以正常抗锯齿设置显示，对于字体大小大于10，正常抗锯齿设置都能显示的很好，但是对于小于10的字体正常抗锯齿就能难表现出好效果了，这时我们可通过文本框的anti-alias type 设为advanced 并且使用*gridFitType* 和sharpness 属性进行精确控制。

TextField.antiAliasType 属性接受一个 *flash.text.AntiAliasType* 常量如 NORMAL（默认）或 ADVANCED。设置为ADVANCED能进行更多精确控制：

```
field.antiAliasType = AntiAliasType.ADVANCED;
```

*gridFitType*属性决定字体外框如何吸附到整个像素，其可用的值是*flash.text.GridFitType*类的NONE, PIXEL,和SUBPIXEL常量。默认为NONE，表示不吸附到整个像素，这样表现为文字比较小时会变得模糊，PIXEL 设置字体外框进行水平和垂直吸附到整个像素，PIXEL 设置只针对文本是左对齐时有效，如果是居中或右对齐，应设置为SUBPIXEL。

```
field.gridFitType = GridTypeFit.PIXEL;
```

sharpness 属性范围在-400 到 400，默认为 0 ，它决定字体外框的清晰程度。值越低越模糊，越高越锐利。当文字比较模糊时可以设置 *gridFitType* 为 PIXEL 或 SUBPIXEL 以提高清晰度。

9.29. 替换文本

问题

我想替换一些文本

解决办法

使用*replaceSelectedText()*方法替换选中的文字或用*replaceText()*方法替换某一范围的文字

讨论

*replaceSelectedText()*方法替换掉正选中的文字，调用该方法前记得先让文本框获得焦点：

```
_field.replaceSelectedText("new text");
```

使用*replaceText()*方法替换掉指定范围的文本，下面的例子替换掉100到150位置的文本：

```
_field.replaceText(100, 150, "new text");
```


9.30. 获取系统字体列表

问题

我想知道用户系统里装了什么字体

解决办法

使用*TextField.fontList* 静态属性

讨论

当我们想使用系统字体时，首选要确定用户系统里装了什么字体，这时可用*TextField.fontList* 属性获得用户系统的字体列表：

```
trace(TextField.fontList);
```

10.0. 简介

在 ActionScript 里可应用多个不同的转换和滤镜处理已达到改变颜色，形状，旋转，大小或显示特殊效果。转换包括颜色，形状，旋转和大小。其他的效果可通过滤镜实现，比如模糊等。

10.1. 改变颜色

问题

我想改变可视化对象的颜色

解决办法

赋值*flash.geom.ColorTransform*对象给可视化对象的transform.colorTransform属性

讨论

每个可视化对象都有一个transform.colorTransform属性，colorTransform属性是一个*flash.geom.ColorTransform*对象实例，它决定对象的颜色。colorTransform属性总是返回*ColorTransform*对象的一个拷贝，也就说不能直接改变colorTransform属性，而是要先获取colorTransform属性，修改，然后再重新赋值给colorTransform属性，像下面这样：

```
var color:ColorTransform = sampleSprite.transform.colorTransform;
color.rgb = 0xFFFFFF;
sampleSprite.transform.colorTransform = color;
```

*ColorTransform*类定义了两种方式修改对象的颜色。一种是直接对rgb属性赋值，该属性接受一个*uint*值，通常为十六进制数(0xRRGGBB)，就像上面的例子那样，还有种方法就是分别设置redOffset, greenOffset, blueOffset,和alphaOffset属性，它们的范围从-255 到255：

```
var color:ColorTransform = sampleSprite.transform.colorTransform;
color.redOffset = 255;
color.greenOffset = 255;
color.blueOffset = 255;
sampleSprite.transform.colorTransform = color;
```

上面的代码通过三原色设置为白色，应用到 sampleSprite 上。如果再改变 alphaOffset 则会出现透明效果，这和 alpha 属性是一样的效果，只不过 alphaOffset 值范围在 0 到 255，而 alpha 值范围在 0 到 100。

10.2.应用色彩

问题

我想改变色彩而不是用纯颜色

解决办法

使用可视化对象的transform.colorTransform属性，但是不是offset属性，而是multiplier属性

讨论

第10.1节 演示如何改变对象的纯色，这会破坏对象的对比度，应用纯色就相当于用同一种颜色填充了整个图形，也就是说对象的每个像素都是同一个颜色。

当应用色彩时只是在每个像素的基础上改变颜色，也就说对象原来的对比度仍然保留。应用色彩和纯色基本上差不多，只不过这回改变的是multiplier属性。

multiplier属性(redMultiplier, greenMultiplier, blueMultiplier, 和alphaMultiplier) 范围都在0到1。multiplier值决定原始像素颜色的倍增值，默认值1，表示增强1倍。下面的例子中绿色被增强1倍，红色和蓝色保持不变：

```
var color:ColorTransform = sampleSprite.transform.colorTransform;
color.redMultiplier = 0;
color.blueMultiplier = 0;
sampleSprite.transform.colorTransform = color;
```


10.3. 重置颜色

问题

我要恢复对象颜色到默认值

解决办法

新建一个*ColorTransform*对象赋值给可视化对象的transform.colorTransform属性

讨论

通过默认值的*flash.geom.ColorTransform*对象来重置可视化对象的颜色，具体的话就是创建一个*ColorTransform*对象直接赋值：

```
sampleSprite.transform.colorTransform = new ColorTransform(    );
```

10.4. 倾斜

问题

我想让一个对象倾斜显示

解决办法

用默认值创建一个*Matrix*对象，其中的b和c属性控制Y方向和X方向的倾斜，然后把*Matrix*对象赋值给可视化对象的transform.matrix属性

讨论

倾斜根据参考点两边方法的所有像素向反方向移动相同距离，这个效果可使一个矩形变成一个平行四边形。

*flash.geom.Matrix*类定义了a，b，c，d，tx，和ty属性。b和c决定倾斜度(a和d决定缩放比，tx和ty决定x和y平移值)。b属性决定Y坐标的倾斜值，c属性决定X坐标的倾斜值，b和c默认为0，值越大越向下和向右倾斜。如果是负数值则向反方向倾斜，下面的代码画出一个矩形，然后向Y方向倾斜，*Matrix*对象使用了默认值，除了b属性，它被设置成1(默认值为 a=1, b=0, c=0, d=1, tx=0, and ty=0)：

```
var box:Sprite = new Sprite(    );
box.graphics.lineStyle(    );
box.graphics.drawRect(0, 0, 100, 100);
addChild(box);
box.transform.matrix = new Matrix(1, 1, 0, 1, 0, 0);
```

10.5. 应用简单的滤镜

问题

我想加入些滤镜效果，如阴影，模糊，光晕，倒角

解决办法

构造一个filter对象，然后赋值给可视化对象的filters数组

讨论

*flash.filters*包包含了下列基础滤镜类：*DropShadowFilter*, *BlurFilter*, *GlowFilter*, *BevelFilter*, *GradientGlowFilter*,和*GradientBevelFilter*。把它们归结为基础的滤镜是因为它们不需要格外添加一些显示对象作为表明映射和矩阵等复杂转换之用。每个滤镜都有一些简单的属性组成，例如*DropShadowFilter*类可以自己设置阴影偏移，颜色，模糊量等属性。

构造好了滤镜对象后就可以赋值给对象的filters属性了。filters属性是一个滤镜对象数组，下面的例子应用了一个阴影滤镜到sampleSprite上：

```
sampleSprite.filters = [new DropShadowFilter(    )];
```

当我们把滤镜数组赋值给可视化对象时，赋值的是数组拷贝而不是引用，因此如果修改了滤镜数组后必须重新赋值，否则无效果，看下面的例子：

```
var dropShadow:DropShadowFilter = new DropShadowFilter(    );
var sampleFilters:Array = [dropShadow];
//应用滤镜
sampleSprite.filters = sampleFilters;
//改变阴影颜色，但是赋值给对象的滤镜拷贝还是原来的颜色，因此需要重新赋值
dropShadow.color = 0xFFFFFF;
//添加光晕效果，需要重新赋值
sampleFilters.push(new GlowFilter(    ));
//重新赋值
sampleSprite.filters = sampleFilters;
```

当我们读取显示对象的filters属性时，它返回滤镜数组的拷贝，这意味着不能用直接对对象的filters属性进行修改：

```
var dropShadow:DropShadowFilter = new DropShadowFilter(    );
// 应用阴影滤镜
sampleSprite.filters = [dropShadow];
// 没有添加光晕滤镜.
```

```
sampleSprite.filters.push(new GlowFilter(    ));
// 正确的做法是取得滤镜数组，添加新滤镜，再重新赋值回去
var sampleFilters:Array = sampleSprite.filters;
sampleFilters.push(new GlowFilter(    ));
sampleSprite.filters = sampleFilters;
```

滤镜效果是一层一层叠加上去的，例如，有个滤镜数组有两个滤镜：阴影滤镜和光晕滤镜（光晕滤镜在第二个位置），当第一个滤镜应用后，第二个滤镜应用在原始对象和第一个滤镜之上。

如果你不想让滤镜产生叠加，必须每个滤镜都应用到对象的拷贝上，思考下下面的例子，光晕滤镜应用到整个表明，包括阴影和原始对象，这不是我们所期望的，我们是希望每个滤镜效果都只影响原始对象而不是产生叠加。

```
var box:Sprite = new Sprite(    );
box.graphics.lineStyle(    );
box.graphics.beginFill(0xFFFFFF);
box.graphics.drawRect(0, 0, 100, 100);
box.graphics.endFill(    );
addChild(box);
box.filters = [new DropShadowFilter(10), new GlowFilter(    )];
```

下面画了两个矩形，分别应用光晕和阴影效果，其中阴影滤镜的knockout属性设为TRue：

```
var box:Sprite = new Sprite(    );
box.graphics.lineStyle(    );
box.graphics.beginFill(0xFFFFFF);
box.graphics.drawRect(0, 0, 100, 100);
box.graphics.endFill(    );
var boxShadow:Sprite = new Sprite(    );
boxShadow.graphics.lineStyle(    );
boxShadow.graphics.beginFill(0xFFFFFF);
boxShadow.graphics.drawRect(0, 0, 100, 100);
boxShadow.graphics.endFill(    );
addChild(boxShadow);
addChild(box);
box.filters = [new GlowFilter(    )];
boxShadow.filters = [new DropShadowFilter(10, 45, 0, 1, 4, 4, 1, 1, false, true)];
```

knockout 属性用于隐藏原始图形，例如，设置阴影滤镜的knockout属性后显示出来只有阴影本身，这一点非常有用，这样我们就能产生不叠加的滤镜组合了。
要想清楚显示对象上的滤镜效果，通过赋值个空数组或null即可：
sampleSprite.filters = [];

10.6. 应用高级滤镜效果(浮雕等)

问题

我要应用高级滤镜效果，如浮雕，边检测等

解决办法

使用*ConvolutionFilter*对象

讨论

*flash.filters.ConvolutionFilter*类可创建从亮度和对比度变换到更多浮雕，模糊，边检测，锐利等动态效果。

联合滤镜需要一个数组定义一个矩阵，通过调整过的像素值把每个象素映射到新的位图表面上(这里面所涉及到的数学理论已超过本书范围)。*ConvolutionFilter*类可做到应用多个效果而不必关心矩阵倍增和如何应用像素映射，我们只需要知道矩阵数组所对应的每个滤镜。

*ConvolutionFilter*构造函数定义所有参数的默认值，但是一旦一个滤镜被应用，则必须定义前三个参数，前两个参数定义矩阵纬数，第三个参数就是矩阵数组。

第一个参数定义列数，第二个参数定义行数，看下面的例子矩阵定义了4列2行：

1　2　3　4

5　6　7　8

虽然可以在任何纬度上定义矩阵，但本书所讨论的都是矩形矩阵（行数和列数都相等的矩阵）。

第三个参数（矩阵数组）指定从左到右，从上到下。例如，下面的矩阵例子可这样用数组表示:

[1, 2, 3, 4, 5, 6, 7, 8]

一个以1为中心周围都是0的矩形矩阵表示没有任何效果，例如：

sampleSprite.filteres = [ConvolutionFilter(3, 3, [0, 0, 0, 0, 1, 0, 0, 0, 0])];

当矩阵数组值之和为1，就像上面的例子那样表示亮度上没有效果，如果之和更高表示亮度被提升，更低表示亮度下降，下面的例子表示显示对象上除了亮度提升没有其他任何效果：

sampleSprite.filteres = [ConvolutionFilter(3, 3, [0, 0, 0, 0, 2, 0, 0, 0, 0])];

下面的例子应用了模糊效果，但是数组值之和大于1会使对象变得更亮，这样的情况下亮度太高已经很难看到加入的模糊效果了：

```
sampleSprite.filters = [new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1, 1, 1, 1, 1])];
```

为了解决亮度问题，我们添加了一个除数作为第四个参数。除数会校正亮度问题而不会影响我们需要的效果(如模糊或锐化)。要重置亮度，把除数设置为数组值之和即可。下面的例子应用了模糊效果而且重置了亮度：

```
sampleSprite.filters = [new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1, 1, 1, 1, 1], 9)];
```

提高中心点的值即可提高模糊效果，不过别忘了提高除数。

10.7. 浮雕

问题

我要应用浮雕效果

解决办法

使用浮雕矩阵创建联合矩阵

讨论

浮雕效果可是对象表面边产生凹凸现象，一个浮雕矩阵中心有个正数，以它为对称轴的的两对数正好是绝对值相等的正负值。他们决定了浮雕的量度，值越大，浮雕效果越明显，中心的上下两个值决定浮雕是凸起还是凹下，下面的例子描述了通用的浮雕矩阵写法：

a d -c

b e -b

c -d -a

下面是一个基本的浮雕矩阵：

-1 1 1

-1 1 1

-1 -1 1

下面的3x3 矩阵是一个比较常用的浮雕矩阵：

-2 -1 0

-1 1 1

0 1 2

下面的例子应用上面的浮雕矩阵，注意这里的除数不需要了：

```
sampleSprite.filters = [new ConvolutionFilter(3, 3, [-2, -1, 0, -1, 1, 1, 0, 1, 2])];
```

10.8. 边检测

问题

我要检测显示对象的边界

解决办法

使用边检测矩阵创建联合滤镜

讨论

用联合滤镜实现边检测效果，使用一个负数为中心的对称矩阵，看下面的写法：

a b c

d e d

c b a

下面是一个通用的的边检测矩阵：

0 1 0

1 -3 1

0 1 0

下面的例子应用了上面的边检测矩阵：

```
sampleSprite.filters = [new ConvolutionFilter(3, 3, [0, 1, 0, 1, -3, 1, 0, 1, 0])];
```

中心数越大，检测到的边越少：

```
sampleSprite.filters = [new ConvolutionFilter(3, 3, [0, 1, 0, 1, -1, 1, 0, 1, 0], 3)];
```

10.9. 锐化

问题

我要应用锐化效果

解决办法

使用锐化矩阵创建联合滤镜

讨论

锐化矩阵和边检测矩阵很类似，只不过边检测矩阵中心是个负数，而锐化矩阵中心是正数，中心周围是负数，下面的例子应用了锐化效果：

```
sampleSprite.filter = [new ConvolutionFilter(3, 3, [0, -1, 0, -1, 5, -1, 0, -1, 0])];
```

提高中心值和除数降低锐化效果：

```
sampleSprite.filter = [new ConvolutionFilter(3, 3, [0, -1, 0, -1, 10, -1, 0, -1, 0], 5)];
```

降低中心值和除数提高锐化效果：

```
sampleSprite.filter = [new ConvolutionFilter(3, 3, [0, -1, 0, -1, 1, -1, 0, -1, 0], -3)];
```

10.10. 制作胶片效果

问题

我要得到显示对象的胶片效果

解决办法

使用胶片矩阵创建*ColorMatrixFilter*对象

讨论

使用胶片矩阵创建*flash.filters.ColorMatrixFilter*对象，胶片矩阵如下：

-1 0 0 0 255

0 -1 0 0 255

0 0 -1 0 255

0 0 0 1 0

下面的例子应用了胶片效果：

```
sampleSprite.filters = [new ColorMatrixFilter([-1, 0, 0, 0, 255, 0, -1, 0, 0, 255, 0, 0, -1, 0, 255, 0, 0, 0, 1, 0])];
```

使用ascb.filters.ColorMatrixArrays.DIGITAL_NEGATIVE常量简单些：

```
sampleSprite.filters = [new ColorMatrixFilter(ColorMatrixArrays.DIGITAL_NEGATIVE)];
```

10.11. 应用灰度效果

问题

我想应用一个灰度效果

解决办法

使用灰度矩阵创建*ColorMatrixFilter*对象

讨论

应用灰度效果可使所有颜色都转换为黑白色，下面是一个灰度矩阵描述：

0.3086　0.6094　0.0820　0　0

0.3086　0.6094　0.0820　0　0

0.3086　0.6094　0.0820　0　0

0　　0　　0　　1　0

下面的例子应用了灰度效果：

```
sampleSprite.filters = [new ColorMatrixFilter([0.3086, 0.6094, 0.0820, 0, 0, 0.3086, 0.6094, 0.0820, 0, 0, 0.3086, 0.6094, 0.0820, 0, 0, 0, 0, 0, 1, 0])];
```

也可使用ascb.filters.ColorMatrixArrays.GRAYSCALE常量：

```
sampleSprite.filters = [new ColorMatrixFilter(ColorMatrixArrays.GRAYSCALE)];
```

10.12. 改变饱和度

问题

我想改变对象的饱和度

解决办法

使用饱和度矩阵创建*ColorMatrixFilter*对象

讨论

饱和度矩阵：

a b c 0 0

d e f 0 0

g h i 0 0

0 0 0 1 0

具体计算公式看下面，i是饱和度值：

a = (1 value) * red + value

b = (1 value) * green

c = (1 value) * blue

d = (1 value) * red

e = (1 value) * green + value

f = (1 value) * blue

g = (1 value) * red

h = (1 value) * green

i = (1 value) * blue + value

当饱和度值为0时，这个矩阵就是灰度矩阵。

我们可使用*ascb.filters.ColorMatrixArrays.getSaturationArray()*方法构造一个饱和度矩阵数组，只需要传递饱和度值即可。

```
sampleSprite.filters = [new ColorMatrixFilter(ColorMatrixArrays.getSaturationArray(2))];
```

10.13. 改变亮度

问题

我想改变对象的亮度

解决办法

使用矩阵创建*ColorMatrixFilter*对象，也可用*ConvolutionFilter*对象改变亮度

讨论

使用矩阵构造一个*ColorMatrixFilter*对象即可调整亮度或者为红，绿，蓝设置偏移值，下面的矩阵是一个通用的矩阵：

a 0 0 0 0

0 a 0 0 0

0 0 a 0 0

0 0 0 1 0

下面的矩阵为红，绿，蓝设置相等的偏移量：

1 0 0 0 a

0 1 0 0 a

0 0 1 0 a

0 0 0 1 0

下面的例子通过增加两倍的颜色来增强亮度：

```
sampleSprite.filters = [new ColorMatrixFilter([2, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 0])];
```

当然也可像第 10.6 节讲到的那样用 *ConvolutionFilter* 对象改变亮度。

10.14. 改变对比度

问题

我想调整对象的对比度

解决办法

使用对比度矩阵创建*ColorMatrixFilter*对象

讨论

和第10.13节类似，通过倍增或偏移颜色值达到调整亮度，对比度也同样如此，下面的矩阵描述了一个通用的对比度矩阵：

a 0 0 0 b

0 a 0 0 b

0 0 a 0 b

0 0 0 1 0

通过下面的公式，我们可以计算出倍增值或偏移值：

a = value * 11

b = 63.5 (value * 698.5)

也可用*ascb.filters.ColorMatrixArrays.getContrastArray()*方法传递一个对比度值构造一个对比度数组，对比度值范围在0到1：

```
sampleSprite.filters = [new ColorMatrixFilter(ColorMatrixArrays.getContrastArray(1))];
```


11.0. 简介

动画的制作形式是非常多样化的，比如随便更改一下可视化对象的外观即可形成一个动画，或者移动一下，改变一下颜色或透明度等等都可制作成动画。

早期版本的Flash中大多数的动画都是通过时间轴来实现的，一个对象被放置在关键帧上，然后设置另一个关键帧，这个关键帧中对对象作一些变化，中间部分通过插值运算形成一个时间轴，也就实现了一段动画。

现在我们所创建的动画都是从电影剪辑(movie clip)或精灵(sprite)开始，它们都包含一个graphics，方法和属性以达到移动，缩放，旋转及其他变换，其中电影剪辑更是提供了原有Flash创作环境，它添加时间线，关键帧。

最后，我们需要把对对象的改变和时间联系起来以实现动画，最好的方式就是通过 *enterFrame* 事件或一个定时器。

11.1. 移动物体

问题

在sprite中有个图形，我想让它动起来

解决办法

先决定x或y轴(或两者)的速率，然后在每一帧中通过速率改变物体的位置

讨论

速率和速度不是同一个概念，速率还包含方向因素，比如说："10 米每小时" 是速度，但是"10米每小时正北方向"是速率。在x或y轴上肯定是考虑方向的，一个正的速率代表x轴的右边，负的为左边。

第一个例子定义了*x*速率：_vx，设置为3，而且例子使用了*enterFrame*事件做动画，在每一帧上对象向右移动3像素：

```
package {
    import flash.display.Sprite;
    import flash.events.Event;
    public class Velocity extends Sprite {
        private var _sprite:Sprite;
        private var _vx:Number = 3;
        public function Velocity(  ) {
            _sprite = new Sprite(  );
            _sprite.graphics.beginFill(0x0000ff, 100);
            _sprite.graphics.drawCircle(0, 0, 25);
            _sprite.graphics.endFill(  );
            _sprite.x = 50;
            _sprite.y = 100;
            addChild(_sprite);
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
        }
        public function onEnterFrame(event:Event):void {
            _sprite.x += _vx;
        }
    }
```

```
}
```

如果设置_vx为-3，你会发现物体在向反方向移动。现在我们再添加y速率：_vy，给它个值，修改下onEnterFrame方法，如下：

```
public function onEnterFrame(event:Event):void {
        _sprite.x += _vx;
        _sprite.y += _vy;
}
```

如果你不喜欢基于帧做动画(上面的例子)，还可以使用定时器函数：

```
package {
        import flash.display.Sprite;
        import flash.events.Event;
        import flash.events.TimerEvent;
        import flash.utils.Timer;
        public class Velocity extends Sprite {
                private var _sprite:Sprite;
                private var _vx:Number = 3;
                private var _vy:Number = 2;
                private var _timer:Timer;
                public function Velocity(    ) {
                        _sprite = new Sprite(    );
                        _sprite.graphics.beginFill(0x0000ff, 100);
                        _sprite.graphics.drawCircle(0, 0, 25);
                        _sprite.graphics.endFill(    );
                        _sprite.x = 50;
                        _sprite.y = 100;
                        addChild(_sprite);
                        _timer = new Timer(30);
                        _timer.addEventListener("timer", onTimer);
                        _timer.start(    );
                }
                public function onTimer(event:TimerEvent):void {
                        _sprite.x += _vx;
```

```
                _sprite.y += _vy;
            }
        }
    }
```

11.2. 在指定方向上移动

问题

我想让物体以一定的速度在指定的方向上移动

解决办法

转换速度和角度为x和y速率，进而改变对象的x和y轴位置

讨论

11.1节 解释了如何在x和y轴上移动物体，但是如果知道了角度和速度，那该怎么移动物体呢？

例如：我想让物体沿着135度移动，速度为每帧4像素。

这个时候我们可利用基本的数学知识把角度和速度转换为x和y速率。首先，我们要确定角度，如果需要弧度，需要下面的公式转换：

```
radians = degrees * Math.PI / 180;
```

如果需要度数，这用这个公式：

```
degrees = radians * 180 / Math.PI;
```

得到弧度后，即可用*Math.sin*和*Math.cos*函数结合速度算出x和y速率了：

```
vx = Math.cos(angle) * speed;
vy = Math.sin(angle) * speed;
```

下面的例子物体沿着135度每帧4像素移动：

```
package {
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.events.TimerEvent;
    import flash.utils.Timer;
    public class AngularVelocity extends Sprite {
```

```
        private var _sprite:Sprite;
        private var _angle:Number = 135;
        private var _speed:Number = 4;
        private var _timer:Timer;
        public function AngularVelocity (    ) {
            _sprite = new Sprite(    );
            _sprite.graphics.beginFill(0x0000ff, 100);
            _sprite.graphics.drawCircle(0, 0, 25);
            _sprite.graphics.endFill(    );
            _sprite.x = 200;
            _sprite.y = 100;
            addChild(_sprite);
            _timer = new Timer(30);
            _timer.addEventListener("timer", onTimer);
            _timer.start(    );
        }
        public function onTimer(event:TimerEvent):void {
            var radians:Number = _angle * Math.PI / 180;
            var vx:Number = Math.cos(radians) * _speed;
            var vy:Number = Math.sin(radians) * _speed;
            _sprite.x += vx;
            _sprite.y += vy;
        }
    }
}
```

11.3. 减速运动

问题

我想让物体平滑的移动到指定位置，就是要慢慢降速，直到停止在指定的位置

解决办法

使用抛物线公式

讨论

首先我们看看减速的概念，察看一个物体从当前位置移动到另一个位置，根据两点之间的距离物体每次移动总距离的1/2，1/3或更少，重复此过程知道物体到达目标位置。

你会发现第一次移动距离很大，接着越来越慢直到停止移动，用速率的观点来说，速率开始时很高，慢慢地减少直到零，另外该速率依赖于距离，距离越大速率越高。

下面的例子掩饰了简单的减速运动。目标位置保存在_targetX和_targetY变量。每次移动比例保存在_easingSpeed变量中，这里设置为0.1，表示总距离的1/10。

```
package {
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.events.TimerEvent;
    import flash.utils.Timer;
    public class Easing extends Sprite {
        private var _sprite:Sprite;
        private var _easingSpeed:Number = 0.1;
        private var _targetX:Number = 400;
        private var _targetY:Number = 200;
        private var _timer:Timer;
        public function Easing(    ) {
            _sprite = new Sprite(    );
            _sprite.graphics.beginFill(0x0000ff, 100);
            _sprite.graphics.drawCircle(0, 0, 25);
            _sprite.graphics.endFill(    );
            _sprite.x = 50;
            _sprite.y = 50;
            addChild(_sprite);
```

```
            _timer = new Timer(30);
            _timer.addEventListener("timer", onTimer);
            _timer.start(    );
        }
        public function onTimer(event:TimerEvent):void {
            var vx:Number = (_targetX - _sprite.x) * _easingSpeed;
            var vy:Number = (_targetY - _sprite.y) * _easingSpeed;
            _sprite.x += vx;
            _sprite.y += vy;
        }
    }
}
```

这里有个小问题，当物体到底目标位置时，定时器却仍然在运行，我们修改下代码，检测距离小于1时停止定时器：

```
public function onTimer(event:TimerEvent):void {
    var dx:Number = _targetX - _sprite.x;
    var dy:Number = _targetY - _sprite.y;
    var dist:Number = Math.sqrt(dx * dx + dy * dy);
    if(dist < 1)
    {
        _sprite.x = _targetX;
        _sprite.y = _targetY;
        _timer.stop(    );
    }
    else
    {
        var vx:Number = dx * _easingSpeed;
        var vy:Number = dy * _easingSpeed;
        _sprite.x += vx;
        _sprite.y += vy;
    }
}
```

上面的例子代码首先计算出两点之间的距离，如果距离小于1，则把物体移到目标位置上，然后停止定时器。

有时候你可能不希望减速到停止位置，比如物体跟随鼠标移动，这里我们只需要把_targetX和_targetY替换为mouseX和mouseY即可：：

```
public function onTimer(event:TimerEvent):void {
    var vx:Number = (mouseX - _sprite.x) * _easingSpeed;
    var vy:Number = (mouseY - _sprite.y) * _easingSpeed;
    _sprite.x += vx;
    _sprite.y += vy;
}
```

11.4. 加速运动

问题

我想让物体加速移动

解决办法

应用加速方法

讨论

许多人认为加速只是简单的提高速度而已，比如想让车开的快些就踩一下加速器。更科学的定义为速率的变化称为加速。虽然大多数情况只是提高物体的速度，实际上还包括减速和改变方向。

下面的例子中变量_ax和_ay代表加速，_vx和_vy代表速率：

```
package {
    import flash.display.Sprite;
    import flash.events.Event;
    public class Accel extends Sprite {
        private var _sprite:Sprite;
        private var _ax:Number = .3;
        private var _ay:Number = .2;
        private var _vx:Number = 0;
        private var _vy:Number = 0;
```

```
        public function Accel(    ) {
            _sprite = new Sprite(    );
            _sprite.graphics.beginFill(0x0000ff, 100);
            _sprite.graphics.drawCircle(0, 0, 25);
            _sprite.graphics.endFill(    );
            _sprite.x = 50;
            _sprite.y = 100;
            addChild(_sprite);
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
        }
        public function onEnterFrame(event:Event):void {
            _vx += _ax;
            _vy += _ay;
            _sprite.x += _vx;
            _sprite.y += _vy;
        }
    }
}
```

除了改变速度，还可以改变方向：

```
var angle:Number = 45;
var accel:Number = .5;
```

转换每个轴上的加速：

```
var radians:Number = angle * Math.PI / 180;
_ax = Math.cos(radians) * accel;
_ay = Math.sin(radians) * accel;
```

把这两个值添加到速率中。

11.5. 弹跳

问题

我想让物体弹跳起来

解决办法

使用*Hooke's定律*---弹簧算法

讨论

Hooke's定律描述了弹簧的运动规律，一般弹簧都有不同的弹力即弹簧所拥有的能量，或大或小，我们用_k变量表示弹簧能量的大小，设为0.1或0.2较好。

ActionScript的弹簧模型还需要个目标点作为物体的弹跳点，另外还需要设置一些阻尼系数，在真实世界里物体弹跳过程会慢慢失去能量，设置能量属性为0.95表示每次弹跳丢失5%的能量，直到物体停止跳动，下面的例子掩饰了弹簧的原理：

```
package {
    import flash.display.Sprite;
    import flash.events.Event;
    public class Spring extends Sprite {
        private var _sprite:Sprite;
        private var _vx:Number = 20;
        private var _vy:Number = 0;
        private var _k:Number = .1;
        private var _damp:Number = .94;
        private var _targetX:Number = 200;
        private var _targetY:Number = 200;
        public function Spring(  ) {
            _sprite = new Sprite(  );
            _sprite.graphics.beginFill(0x0000ff, 100);
            _sprite.graphics.drawCircle(0, 0, 25);
            _sprite.graphics.endFill(  );
            _sprite.x = 0;
            _sprite.y = 0;
            addChild(_sprite);
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
```

```
        }
        public function onEnterFrame(event:Event):void {
            var ax:Number = (_targetX - _sprite.x) * _k;
            var ay:Number = (_targetY - _sprite.y) * _k;
            _vx += ax;
            _vy += ay;
            _sprite.x += _vx;
            _sprite.y += _vy;
            _vx *= _damp;
            _vy *= _damp;
        }
    }
}
```

把目标点改成鼠标坐标试试：

```
var ax:Number = (mouseX - _sprite.x) * _k;
var ay:Number = (mouseY - _sprite.y) * _k;
```

11.6. 使用三角定理

问题

我想做一些高级动画，如旋转，循环运动或摆动

解决办法

使用内建的函数：*Math.sin()*，*Math.cos()*，和*Math.atan2()*.

讨论

11.2节和11.4节 已经使用了正弦和余弦函数，除此它们还被用来产生更有用的效果，比如围绕一个中心或沿着直线做运动，旋转等效果。*Math.sin()*和*Math.cos()*函数都是基于正三角形（有一个角等于90度）。当增大函数的参数数值时，函数的返回值从-1到0, 1, 0, 回到-1, 如下代码所示：

```
for(var i:Number = 0; i < 10; i += 0.1) {
    trace(Math.sin(i));
}
```

上面的代码输出一串数字，从0开始，到0.999，再到-0.999，再试着增大这个数字，比如40，那就会输出-40到40。利用正弦我们就能做出摆动效果，如下例子：

```
package {
    import flash.display.Sprite;
    import flash.events.Event;
    public class Oscillation extends Sprite {
        private var _sprite:Sprite;
        private var _angle:Number = 0;
        private var _radius:Number = 100;
        public function AS3CB(  ) {
            _sprite = new Sprite(  );
            _sprite.graphics.beginFill(0x0000ff, 100);
            _sprite.graphics.drawCircle(0, 0, 25);
            _sprite.graphics.endFill(  );
            _sprite.x = 0;
            _sprite.y = 100;
            addChild(_sprite);
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
```

```
            }
            public function onEnterFrame(event:Event):void {
                _sprite.x = 200 + Math.sin(_angle) * _radius;
                _angle += .05;
            }
        }
}
```

这里的_angle变量保持增大，然后作为*Math.sin()*的参数，输出结果在乘以_radius变量，它被设为100，这样结果就是物体向左右摆动100像素。

*Math.cos()*也能达到同样的效果，两个叠加形成圆周运动：

```
public function onEnterFrame(event:Event):void {
    _sprite.x = 200 + Math.sin(_angle) * _radius;
    _sprite.y = 200 + Math.cos(_angle) * _radius;
    _angle += .05;
}
```

如果要形成椭圆形运动，这设置不同的弧度值，如设置_xRadius为100，_yRadius为50：

```
public function onEnterFrame(event:Event):void {
    _sprite.x = 200 + Math.sin(_angle) * _xRadius;
    _sprite.y = 200 + Math.cos(_angle) * _yRadius;
    _angle += .05;
}
```

如果要实现看似随即的运动轨迹的话，可独立设置每个轴的相关变量：

```
private var _xAngle:Number = 0;
private var _yAngle:Number = 0;
private var _xSpeed:Number = .13;
private var _ySpeed:Number = .09;
private var _xRadius:Number = 100;
private var _yRadius:Number = 50;
Then apply those to the motion code:
public function onEnterFrame(event:Event):void {
    _sprite.x = 200 + Math.sin(_xAngle) * _xRadius;
    _sprite.y = 200 + Math.cos(_yAngle) * _yRadius;
```

```
        _xAngle += _xSpeed;
        _yAngle += _ySpeed;
    }
```

另一个有用的函数是*Math.atan2()*。主要用它来计算出两点的夹角，它接受两个参数，两点在y轴上的距离和在x轴上的距离，然会返回角度值。

下面的例子创建了“跟随眼睛”效果，桌面的小工具，它会一直看着鼠标：

```
package {
    import flash.display.Sprite;
    import flash.events.Event;
    public class FollowingEye extends Sprite {
        private var _sprite:Sprite;
        public function AS3CB(    ) {
            _sprite = new Sprite(    );
            _sprite.graphics.beginFill(0xffffff, 100);
            _sprite.graphics.drawCircle(0, 0, 25);
            _sprite.graphics.endFill(    );
            _sprite.graphics.beginFill(0x000000, 100);
            _sprite.graphics.drawCircle(20, 0, 5);
            _sprite.graphics.endFill(    );
            _sprite.x = 100;
            _sprite.y = 100;
            addChild(_sprite);
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
        }
        public function onEnterFrame(event:Event):void {
            var dx:Number = mouseX - _sprite.x;
            var dy:Number = mouseY - _sprite.y;
            var radians:Number = Math.atan2(dy, dx);
            _sprite.rotation = radians * 180 / Math.PI;
        }
    }
}
```

11.7. 运用动画技术

问题

我想把这章里的动画技术应用到对象的运动上

解决办法

运用这些技术，把结果赋值给对象的x和y属性

讨论

虽然改变对象的位置有各种各样的方法，大多数这章讨论的方法都可以被应用到一个电影剪辑或sprite的任何属性上。

首先尝试应用速率到对象的rotation属性上，变量名为_vr：

```
package {
    import flash.display.Sprite;
    import flash.events.Event;
    public class AnimatingRotation extends Sprite {
        private var _sprite:Sprite;
        private var _vr:Number = 4;
        public function AS3CB(    ) {
            _sprite = new Sprite(    );
            _sprite.graphics.beginFill(0xffffff, 100);
            _sprite.graphics.drawRect(-50, -20, 100, 40);
            _sprite.graphics.endFill(    );
            _sprite.x = 100;
            _sprite.y = 100;
            addChild(_sprite);
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
        }
        public function onEnterFrame(event:Event):void {
             _sprite.rotation += _vr;
        }
    }
}
```

我们看到一个矩形在做旋转运动，每帧sprite的rotation增加4。

下面的例子应用弹簧原理缩放sprite。click处理函数产生随即的缩放值，*enterFrame*函数做弹簧运动，点击sprite会产生新的大小：

```
package {
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.events.MouseEvent;
    public class AnimatingProperties extends Sprite {
        private var _sprite:Sprite;
        private var _k:Number = 0.1;
        private var _damp:Number = 0.9;
        private var _scaleVel:Number = 0;
        private var _targetScale:Number = 1;
        public function AS3CB(  ) {
            _sprite = new Sprite(  );
            _sprite.graphics.beginFill(0xffffff, 100);
            _sprite.graphics.drawRect(-50, -50, 100, 100);
            _sprite.graphics.endFill(  );
            _sprite.x = 100;
            _sprite.y = 100;
            addChild(_sprite);
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
            _sprite.addEventListener(MouseEvent.CLICK, onClick)
        }
        public function onEnterFrame(event:Event):void {
            _scaleVel += (_targetScale - _sprite.scaleX) * _k
            _sprite.scaleX += _scaleVel;
            _sprite.scaleY = _sprite.scaleX;
            _scaleVel *= _damp;
        }
        public function onClick(event:MouseEvent):void {
            _targetScale = Math.random(  ) * 2 - .5;
```

```
        }
    }
}
```

下面的例子设置了两组颜色值：_red1, _green1, _blue1，和 _red2, _green2, _blue2。每个值都在0.0到1.0之间。在*enterFrame*处理函数中用这些值设置*color transform* 对象并应用到sprite上：

```
package {
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.events.MouseEvent;
    import flash.geom.ColorTransform;
    public class AnimatingColor extends Sprite {
        private var _sprite:Sprite;
        private var _red1:Number = 1;
        private var _green1:Number = 0;
        private var _blue1:Number = 0;
        private var _red2:Number = 0;
        private var _green2:Number = .5;
        private var _blue2:Number = 1;
        private var _easingSpeed:Number = 0.05;
        public function AS3CB(  ) {
            _sprite = new Sprite(  );
            _sprite.graphics.beginFill(0xffffff, 100);
            _sprite.graphics.drawRect(-50, -50, 100, 100);
            _sprite.graphics.endFill(  );
            _sprite.x = 100;
            _sprite.y = 100;
            addChild(_sprite);
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
            addEventListener(MouseEvent.CLICK, onClick);
        }
        public function onEnterFrame(event:Event):void {
            _red1 += (_red2 - _red1) * _easingSpeed;
```

```
            _green1 += (_green2 - _green1) * _easingSpeed;
            _blue1 += (_blue2 - _blue1) * _easingSpeed;
            _sprite.transform.colorTransform =
                    new ColorTransform(_red1, _green1, _blue1);
        }
        public function onClick(event:MouseEvent):void {
            _red2 = Math.random(    );
            _green2 = Math.random(    );
            _blue2 = Math.random(    );
        }
    }
}
```


12.0. 简介

在ActionScript里字符串是最基本的字符存储类型。一个字符串由双引号或单引号包围的零个或多个字符组成。和其他语言不同的是在ActionScript里单引号和双引号是没有区别的，例如：

```
var exampleA:String = "this is a string";
var exampleB:String = 'this is also a string';
var exampleC:String = "strings can contain characters such as (*+5~";
var exampleD:String = ""; //  空字符串
var exampleE:String = "x"; //单个字符
var exampleF:String; //  默认为null
```

字符串必须在引号之内，而且单引号和双引号不能混用，下面的两种写法都是错误的：

```
var exampleA:String = "an incorrect string';          //  必须以双引号结尾
var exampleB:String = 'another incorrect string"; //  必须以单引号结尾
```

ActionScript 提供了很多方法进行字符串的处理计算，在 ActionScript 3.0 增加了支持字符串的正则表达式(模式匹配)，具体内容请看第十三章。

12.1. 字符串连接

问题

我想把零散的多个字符串连接成一个

解决办法

使用连接操作符+，或者简写成+=，或者使用*String.concat()*方法

讨论

使用+操作符可把多个字符串连接成一个字符串：

```
// 连接成的字符串为"Thisworks" (中间没有空格)
var example:String = "This" + "works";
```

可一次连接多个字符串：

```
// 结果为"This works" (中间有空格)
var example:String = "This" + " " + "works";
```

还可以连接其他类型的数据（自动转换为字符串），例如：

```
var attendance:int = 24;
// 结果为"There are 24 people"
var output:String = "There are " + attendance + " people";
```

+操作符会在连接之前把其他类型的数据转换为字符串后再进行连接，上面的例子把整型的24转换为字符串进行了连接，但是如果所有的待连接的数据都是数字，那编译器就会报错：

```
var first:int = 24;
var second:int = 42;
// 结果编译器报错"Implicit coercion of a value
// type 'Number' to an unrelated type 'String'"
var result:String = first + second;
```

这时有个技巧就是在前面加个空字符串：

```
var first:int = 24;
var second:int = 42;
// 结果为"2442"
var result:String = "" + first + second;
```

而且空字符串必须放在表达式的最前面，这样后面的都将被转换为字符串：

```
var first:int = 24;
var second:int = 42;
```

```
// 结果为"66"
var result:String = first + second + "";
```

另一个方法就是使用*String()*转换函数(强制类型转换)：

```
var first:int = 24;
var second:int = 42;
//结果为"2442"
var result:String = String( first ) + second;
```

而且要转换的必须是表达式的最前面的那个变量，下面的例子结果是不正确的：

```
var first:int = 24;
var second:int = 42;
var third:int = 21;
// 结果为"6621"
var result:String = first + second + String( third );
```

另一个方法就是直接使用对象的*toString()*方法，*Number*和*int*数据类型都有*toString()*方法：

```
var first:int = 24;
var second:int = 42;
var third:int = 21;
//结果为"244221"
var result:String = first.toString() + second + third;
```

如果想在连接的表达式中进行加法运算，加法表达式必须放在括号中：

```
var first:int = 24;
var second:int = 42;
//结果为"There are 66 people"
var result:String = "There are " + ( first + second ) + " people";
```

通过+= 操作符可追加字符串：

```
var attendance:int = 24;
var example:String = "There are ";
example += attendance;
// 结果为"There are 24 people"
example += " people";
```

这种技术很友好好处，比如你的字符串过长，即可用这种方法把字符串分成多个字串，然后逐

个追加上去，从代码上看更容易阅读：

```
var example:String = "This is the first sentence in a long paragraph of text.";
example += "By adding line by line to the string variable you make ";
example += "your code more readable.";
```

你也可以这样写：

```
var example:String = "This is the first sentence in a long paragraph of text. "
    + "By splitting the long string into smaller, more manageable pieces "
    + "and using the + operator, you can make your code more readable.";
```

字符串连接最常见的可能就是聊天程序了，比如聊天程序的历史纪录，聊天内容通过字符串连接会不断地追加历史纪录中，例如：

```
// 下面的方法追加姓名和消息到聊天历史纪录中
private function updateChatHistory( message:String, username:String ):void {
  // 历史纪录：history 变量
  _history += username + ":" + message + '\n';
};
```

通过*String.concat()*方法也可以把新字符串追加到指定的字符串中，该方法不会影响源字符串，而是返回一个新的字符串作为连接结果：

```
var original:String = "original string value.";
// 设置modified变量保存连接结果"original string value.now modified."
// 源original保持不变
var modified:String = original.concat( "now modified." );
```

12.2. 在字符串中使用引号和省略号

问题

我想把引号和省略号作为字符串值

解决办法

使用反斜杆进行转义处理，或者在双引号里使用单引号

讨论

ActionScript 编译器通过双引号或单引号来分析字符串，字符串从引号开始到引号结束，如果中间再出现引号则编译器就会报错，这导致引号不匹配了。

按照编译器的检测来说它不知道哪个引号才是结束标志，下面的例子中，这样嵌套双引号是不正确的：

```
var error:String = "He said, "Yes."";   // 错误写法.
```

一种可行的办法是外围用单引号，内部用双引号区分开来，如下所示：

```
// 表达式正确，结果为He said, "Yes."
var exampleA:String = 'He said, "Yes."';
//反过来也可以，结果为 He said, 'Yes.'
var exampleB:String = "He said, 'Yes.'";
```

但是如果内外都用双引号，这时就只能用通过转义了，通过转义符（\），只要在需要转义的字符前加上它即可：

```
// 结果为：He said, "Yes."
var sExample:String = "He said, \"Yes.\"";
```

转义符告诉编译器后面的字符不代表任何意思，只是原样输出。

12.3. 插入特殊的空格字符

问题

我想在字符串中添加空格字符，比如制表符或新行字符

解决办法

使用特殊字符的转义序列

讨论

下面的表格列出了五种空格字符的转义字符。

可以在字符串里用这些转义序列，这在文本框中显示文本是很有用的：

```
var example:String = "these\twords\tare\tseparated\tby\ttabs";
// 结果为：these	words	are	separated	by	tabs
```

空格字符	转义序列
Newline	\n
Tab	\t
Backspace	\b
Form feed	\f
Carriage return	\r


```
var example:String = "these\nwords\nare\nseparated\nby\nnewlines";
/* 结果为：
these
words
are
separated
by
newlines
*/
```

在ActionScript 3.0 已经取消早期版本所支持的newline变量，代替的是\\n转义序列。

```
// 编译器错误，须用\n代替newline
var error:String = "two" + newline + "lines";
```

在 Flash 里 newline, form feed, 和 carriage return 字符返回的结果一样，当 Flash 载入外部资源时，有些可能含有 newline 字符，有些会是 form feeds，或者 carriage returns。

12.4. 搜索字串

问题

我想在字符串里找出指定的子串

解决办法

使用*String*类的*indexOf()*或*lastIndexOf()*方法

讨论

使用*indexOf()*和*lastIndexOf()*方法可检测出字符串中是否包含指定的子串，每个方法返回匹配子串的起始索引。*indexOf()*方法从左到右搜索，而*lastIndexOf()*方法从右到左搜索，如果没找到则返回-1。

*indexOf()*方法接受两个参数：

substring

指定要搜索的子串

startIndex

可选参数，表示起始搜索位置，默认从0开始。

如果要测试是否一个字符串包含另一个字符串，只需要传入一个参数给*indexOf()*方法即可：

```
var example:String = "This string contains the word cool twice. Very cool.";
// 获得子串"cool"在example中的索引
var index:int = example.indexOf( "cool" );
// 如果indexOf( )方法返回-1，表示没有找到"cool"
if ( index != -1 ) {
  // 显示："String contains word cool at index 30"
  TRace( "String contains word cool at index " + index );
}
```

通过指定第二个参数指定搜索的起始位置，再次搜索，找出第二个匹配的字串位置：

```
var example:String = "This string contains the word cool twice. Very cool.";
// 得到第一个匹配的子串
var index:int = example.indexOf( "cool" );
```

```
if ( index != -1 ) {
   // 显示：  "String contains word cool at index 30"
   trace( "String contains word cool at index " + index );
}
// 得到第二个匹配的子串"cool"
// 传递index + 1，以过滤掉第一个匹配子串，如果只传递index那么返回的仍是第一个子串位置
index = example.indexOf( "cool", index + 1 );
if ( index != -1 ) {
   // 显示："String contains word cool at index 47"
   trace( "String contains word cool at index " + index );
}
```

在while循环语句中使用*indexOf()*可找出所有匹配的子串位置，例如：

```
var example:String = "This string contains the word cool. Very cool. Yes, cool.";
var index:int = -1;
// 循环直到indexOf()返回-1
while ( ( index = example.indexOf( "cool", index + 1 ) ) != -1 ) {
   /* 显示：
        String contains word cool at index 30
        String contains word cool at index 41
        String contains word cool at index 52
   */
   trace( "String contains word cool at index " + index );
}
```

*lastIndexOf()*方法与*indexOf()*类似，只是它是从右边开始匹配搜索，返回第一个匹配的子串索引。

*lastIndexOf()*方法也接受两个参数意义和*indexOf()*相同：

如果搜不到返回-1，默认从字符串的末尾（字符串长度）开始搜索。

```
var example:String = "This string contains the word cool twice. Very cool.";
//得到最后面的那个"cool"索引
var index:int =    example.lastIndexOf( "cool" );
if ( index != -1 ) {
   // 显示: "String contains word cool at index 47"
```

```
    trace( "String contains word cool at index " + index );
}
// 得到第二个"cool"索引
// 通过传递index-1作为第二个参数
index = example.lastIndexOf( "cool", index - 1 );
if ( index != -1 ) {
    // 显示: "String contains word cool at index 30" because the next to last
    trace( "String contains word cool at index " + index );
}
```

和*indexOf()*一样，在while循环里通过*lastIndexOf()*找出所有匹配子串

```
var example:String = "This string contains the word cool. Very cool. Yes, cool.";
var index:int =    example.length;
// 循环直到lastIndexOf(   )返回-1.
while ( ( index = example.lastIndexOf( "cool", index - 1 ) ) != -1 ) {
    /*  显示:
        String contains word cool at index 52
        String contains word cool at index 41
        String contains word cool at index 30
    */
    trace( "String contains word cool at index " + index );
}
```

这里的代码和上面的*indexOf()*很类似，只不过这里的index初始值为example.length而不是-1，因为*lastIndexOf()*是从右边开始搜索，第二个不同点是*lastIndexOf()*第二个参数不是index+1。

*indexOf()*和*lastIndexOf()*方法都是区分大小写的，比如搜索"cool"就不能匹配"Cool"因为大小写不同，要进行不区分大小写搜索，可使用*toLowerCase()*方法进行转换。

```
var example:String = "Cool. This is a string with the word cool. It spells"
                        + " cool as both cool (lowercase) and Cool (capitalized).";
var search:String = "cool";
//输出第一个匹配的"cool"，结果为37,
//因为第一个"Cool"是大写
trace( example.indexOf( search ) );
// 经过小写转换后再次搜索匹配，返回结果为0
```

```
trace( example.toLowerCase(    ).indexOf( search ) );
// 输出最后一个匹配的"cool"，结果为66,
trace( example.lastIndexOf( search ) );
// 经过小写转换后输出为87
trace( example.toLowerCase(    ).lastIndexOf( search ) );
// 现在改变搜索子串为"Cool"
search = "Cool";
// 进行小写转换后输出结果为-1，因为已经没有大写"Cool"存在了
trace( example.toLowerCase(    ).indexOf( search ) );
```

12.5. 获取子串

问题

我要从字符串中提取子串

解决办法

使用*substring()*，*substr()*，或*slice()*方法

讨论

substring()，*substr()*，和*slice()*方法都能返回子串且不会影响原始字符串，不同点是接受参数不同。

*substr()*方法接受两个参数

startIndex

字串第一个字符位置，该值可以是负数，负数表示从字符串末尾开始，-1表示最后一个字符。

length

获取子串的长度，如果忽略此参数表示从起始位置开始到最后。

```
var example:String = "Bunnies";
trace( example.substr( 0 ) );          // 显示：Bunnies
trace( example.substr( 0, 3 ) );     // 显示：Bun
trace( example.substr( 3, 3 ) );     // 显示：nie
trace( example.substr( -1 ) );        // 显示：s
```

```
trace( example.substr( -2, 5 ) );   // 显示：es
```

*substring()*和*slice()*方法接受同样的参数

startIndex

字串第一个字符位置

endIndex

子串的最后个字符位置。子串包含起始位置的字符，但是不包含终止位置的字符，如果此参数忽略，表示从起始位置到源字符串最后。

*substring()*只接受正数，如果遇到负数都被转换为0处理，如果*endIndex*小于*startIndex*,，则会自动对调，因此*startIndex*总是小的数。*slice()*方法的*startIndex*和*endIndex*参数也都是正数，如果*endIndex*小于*startIndex*则返回-1。

```
var example:String = "Rabbits";
// 两者都输出整个字符串，从0开始到最后(example.length - 1).
trace( example.substring( 0 ) );          // 显示: Rabbits
trace( example.slice( 0 ) );              // 显示: Rabbits
// substring()方法没有输出，因为它把负数都转为0了。slice()方法输出"it"，其中不包含最后个字符。
trace( example.substring( -3, -1 ) );   // 无输出
trace( example.slice( -3, -1 ) );         // 显示：it
// 都输出"ab".
trace( example.substring( 1, 3 ) );       // 显示：ab
trace( example.slice( 1, 3 ) );           // 显示：ab
// substring()方法输出子串"ab"，因为它会自动对调两个参数，slice()方法没有输出
trace( example.substring( 3, 1 ) );       // 显示：ab
trace( example.slice( 3, 1 ) );           // 无输出
```

这些函数一般都和*indexOf()*和*lastIndexOf()*方法一起使用，先用*indexOf()*和*lastIndexOf()* 方法搜索子串，然后用上面的这些方法提取子串。

下面的例子提取文件扩展名和文件名（除去扩展名），通过“.”符号分离：

```
var filename:String = "document.jpg";
// 找出“.”符号的位置
var extensionIndex:Number = filename.lastIndexOf( '.' );
// 提取文件名
var extensionless:String = filename.substr( 0, extensionIndex );
trace( "The filename is " + extensionless );
```

```
// 提取扩展名("jpg")
var extension:String = filename.substr( extensionIndex + 1, filename.length );
trace( "The file extension is " + extension );
```

也可用*split()*方法：

```
var filename:String = "document.jpg";
var nameParts:Array = filename.split(".");
var extensionless:String = nameParts[0];
trace ("The filename is " + extensionless);
var extension:String = nameParts[1];
trace ("The file extension is " + extension);
```

这里有个问题，如果文件名本身也含有“.”符号，这么上面的两个方法提取的子串就会不正确，这时候用lastIdexOf()方法比较好：

```
private function removeExtension( filename:String ):String {
   // 找出最后面的“.”字符
   var extensionIndex:Number = filename.lastIndexOf( '.' );
   if ( extensionIndex == -1 ) {
      // 如果没有“.”字符则直接返回filename
      return filename;
   } else {
      return filename.substr( 0, extensionIndex );
   }
}
// 写在函数里可随时调用
private function extractExtension( filename:String ):String {
   var extensionIndex:Number = filename.lastIndexOf( '.' );
   if ( extensionIndex == -1 ) {
      return "";
   } else {
      return filename.substr( extensionIndex + 1, filename.length );
   }
}
```

例子：

```
trace( removeExtension( "document.jpg" ) );          // 显示: document
trace( removeExtension( "document" ) );               // 显示: document
trace( removeExtension( "document.1.jpg" ) );        // 显示: document.1
trace( extractExtension( "document.jpg" ) );         // 显示: .jpg
trace( extractExtension( "document" ) );              // 显示nothing
trace( extractExtension( "document.1.jpg" ) );       // 显示: .jpg
```

12.6. 单词分析

问题

我想处理字符串中的单词

解决办法

使用*split()*方法

讨论

*split()*方法根据指定的分隔符把分离的子串存入数组，要把字符串分离出单词，可空格作为分隔符。

```
// 创建含有多个单词的字符串
var example:String = "This is a string of words";
//用空格作为分隔符分离字符串
var words:Array = example.split( " " );
//循环数组处理每个单词
for ( var i:int = 0; i < words.length; i++ ) {
   /* 显示:
        this
        is
        a
        string
        of
```

```
        words
    */
    trace( words[i] );
}
```

处理单词有很多方法，下面的完整例子演示分离字符串及显示单词：

```
 package {
    import flash.display.Sprite;
    import flash.events.MouseEvent;
    import flash.text.TextField;
    import flash.text.TextFieldAutoSize;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;
    public class ActionScriptPoetry extends Sprite {
        public function ActionScriptPoetry(    ) {
            stage.align = StageAlign.TOP_LEFT;
            stage.scaleMode = StageScaleMode.NO_SCALE;
            // 创建字符串
            var example:String = "This is a string of ActionScript poetry words";
            var words:Array = example.split(" ");
            var word:Sprite;
            var wordText:TextField;
            //遍历每个单词
            for ( var i:int = 0; i < words.length; i++ ) {
                // 为每个单词创建sprite并显示在屏幕上
                word = new Sprite(    );
                addChild( word );
                // 创建一个文本框
                wordText = new TextField(    );
                word.addChild( wordText );
                // 设置文本框大小，边框，背景颜色
                wordText.autoSize      = TextFieldAutoSize.LEFT; // Left-justify the text
```

```
            wordText.border         = true;
            wordText.background = true;
            wordText.selectable = false;
            // 设置每个文本框显示一个单词
            wordText.text = words[i];
            // 点击sprite可拖动
            word.addEventListener( MouseEvent.MOUSE_DOWN, handleDrag );
            word.addEventListener( MouseEvent.MOUSE_UP, handleDrop );
            // sprites的位置随机
            var rx:Number = Math.random(    ) * stage.stageWidth    - word.width;
            var ry:Number = Math.random(    ) * stage.stageHeight - word.height;
            word.x = rx;
            word.y = ry;
            trace(word);
        }
    }
    // 当用户鼠标在单词上点击时调用该函数
    private function handleDrag( event:MouseEvent ):void {
        // 当前事件目标为TextField，因此它的父对象为sprite
        var word:Sprite = event.target.parent;
        // 让单词置顶显示
        setChildIndex( word, numChildren - 1 );
        // 让单词可拖动
        word.startDrag(    );
    }
    // 当松开鼠标时调用该函数
    private function handleDrop( event:MouseEvent ):void {
        // 事件目标为TextField，其父对象为Sprite
        var word:Sprite = event.target.parent;
        // 停止拖动
        word.stopDrag(    );
```

```
    }
  }
}
```

上面的代码使用*split()*函数以空格作为分隔符分离单词，在这里原始字符串还不包括标点符号，如果含有标点符号或其他特殊符号则需用正则表达式作为split的参数过滤掉这些符号才能正确分离出单词，比如下面的正则表达式匹配可以清除掉单词周围的标点符号/[^a-zA-Z0-9]+/.

```
// 创建有标点符号的字符串.
var example:String = "Here are some words. Also, here is some punctuation!";
// 用空格分离出单词，这些单词还含有标点符号
var words:Array = example.split( ' ' );
// 显示这些单词，他们还包含标点符号
for ( var i:int = 0; i < words.length; i++ ) {
  /* Outputs:
      Here
      are
      some
      words.
      Also,
      here
      is
      some
      punctuation!
  */
  trace( words[i] );
}
// 使用正则表达式分离出正确的单词
words = example.split( /[^a-zA-Z0-9]+/ );
// 再次输出这些单词看看，现在已经没有标点符号了
for ( i = 0; i < words.length; i++ ) {
  /* Outputs:
      Here
      are
```

```
        some
        words
        Also
        here
        is
        some
        punctuation
    */
    trace( words[i] );
}
```

关于正则表达式，请看第十三章。

12.7. 删除或替换字符或单词

问题

我要删除字符串上的字符或替换它

解决办法

使用*replace()*方法或*split()*和*join()*

讨论

ActionScript 3.0 提供一个新方法*String.replace()*，用于字符替换，该方法接受两个参数：

pattern

查找或替换的字符串或正则表达式。

replace

替换的字符串，也可以是一个返回字符串的函数。

如果只提供一个参数时该方法有两个用途，这一节讨论字符串模式，正则表达式模式将在 13.4 节讨论。

这里有个简单的替换句子中的子串例子。*replace()*方法返回替换掉的新字符串，原始字符串仍未被修改。

```
var example:String = "This is a cool sentence."
//" is "  替换为" is not "
```

```
// 显示为：This is not a cool sentence.
trace( example.replace( " is ", " is not " ) );
```

在这个例子中替换字符串为" is ",周围是有空格的，这点很重要，否则的话句子中的"This is"也会被替换为"This not is,"，这就不是我们需要的结果了。

用字符串作为匹配模式有两个比较大的问题：替换次数和大小写问题，这些都要自己去处理，如通过循环来替换掉所有匹配子串：

```
//创建字符串，一个原始的，一个为替换掉的
var example:String = "It's a bird, it's a plane, it's ActionScript Man!";
var replaced:String = example; // 先初始化为原始字符串
// 通过循环找出所有的"it's"子串，替换掉
while ( replaced.indexOf( "it's" ) != -1 ) {
   // 替换"it's"为"it is".
   replaced = replaced.replace( "it's", "it is" );
}
// 解决大小写问题
replaced = replaced.replace( "It's", "It is" );
// 最后输出为：It is a bird, it is a plane, it is ActionScript Man!
trace( replaced );
```

*split()*方法也可以被用来替换或删除字符串中的字符或单词。不像*replace()*方法把字符串作为匹配模式，*split*是替换所有的符号，但是这两个方法都是忽略大小写问题。下面的例子用\\n替换掉
：

```
var example:String = "This is<br>a sentence<br>on 3 lines";
/* 显示：
    This is
    a sentence
    on 3 lines
*/
trace( example.split( "<br>" ).join( '\n' ) );
```

split方法返回分离出的字符串数组，而*Array.join()*方法把数组元素有重新组成一个新字符串Split方法删除了分隔符，而*join()*方法正好相反，重新加入新的分隔符。

删除字符或单词就很简单了，只要替换为空串即可：

```
var example:String = "This is a cool sentence.";
```

```
// 删除单词"cool"
// 显示：This is a sentence.
trace( example.replace( "cool ", "" ) );
```

12.8. 每次只读取一个字符

问题

我想每次只读取字符串中的一个字符

解决办法

在*for*语句中使用*String.charAt()*方法，也可以用*String.split()*方法，以空字符串作为分隔符把所有的字符分离出来作为数组，然后再用*for*语句遍历数组。

讨论

最简单的方法就是在for循环中通过字符串的字符下标依次读取每个字符，下标范围为0到string.length-1，使用*charAt()*方法即可读取字符进行处理。

```
var example:String = "a string";
//循环遍历字符串中的每个字符
for ( var i:int = 0; i < example.length; i++ ) {
   /* 每次输出一个字符：
        a
        s
        t
        r
        i
        n
        g
   */
   trace( example.charAt( i ) );
}
```

使用*split()* 方法也可以达到同样的效果，这里我们使用空串作为*split()* 方法的分隔符参数。

```
var example:String = "a string";
```

```
var characters:Array = example.split( "" );
for ( var i:int = 0; i < characters.length; i++ ) {
/*     a

       s
       t
       r
       i
       n
       g
   */
   trace( characters[i] );
}
```

两种方法都可以，不过如果你想要把字符排序一下的话最好用split方法，*charAt()* 方法就没那么容易做到了：

```
var example:String = "a string";
var characters:Array = example.split( "" );
//对字符数组进行排序
characters.sort(    );
for ( var i:int = 0; i < characters.length; i++) {
   /*  显示:

       a
       g
       i
       n
       r
       s
       t
   */
   trace( characters[i] );
}
```

如果想删除某些字符，split方法也比charAt()容易做到：

```
var example:String = "a string";
var characters:Array = example.split( "" );
for ( var i:Number = 0; i < characters.length; i++ ) {
   // 删除所有“r”元素Remove all "r" elements from the array. Be sure to decrement i if an
   // element is removed; otherwise, the next element is improperly skipped.
   if ( characters[i] == "r") {
      characters.splice( i, 1 );
      i--;
   }
}
// 显示: a sting
trace( characters.join( "" ) );
```


12.9. 大小写转换

问题

我想进行字符串的大小写转换以便执行大小写无关的比较运算

解决办法

使用*UpperCase()* 和 *toLowerCase()* 方法。

讨论

*toUpperCase()*和*toLowerCase()* 方法进行大小写处理后返回新的字符串，原始字符串还是未作修改，在进行大小写无关性字符串搜索时这点是很有用的：

```
var example:String = "What case?";
// 显示: what case?
trace( example.toLowerCase(    ) );
// 显示: WHAT CASE?
trace( example.toUpperCase(    ) );
// 原始字符串还是未改变：What case?
trace( example );
```

两个方法都返回新的字符串：

```
var example:String = example.toLowerCase(    );
```

使用*toLowerCase()*和*toUpperCase()* 可以把一个单词的首字母变成大写，其他小写。自定义类*ascb.util.StringUtilities.toInitialCap()*方法就是这个作用，此方法的代码：

```
public static function toInitialCap( original:String ):String {
   return original.charAt( 0 ).toUpperCase(    ) + original.substr( 1 ).toLowerCase(    );
}
```

看下面的例子：

```
var example:String = "bRuCE";
trace( StringUtilities.toInitialCap( example ) );       // 显示: Bruce
```

*toTitleCase()*方法把句子的所有单词首字母变成大写，方法定义如下：

```
public static function toTitleCase( original:String ):String {
   var words:Array = original.split( " " );
   for (var i:int = 0; i < words.length; i++) {
      words[i] = toInitialCap( words[i] );
   }
   return ( words.join( " " ) );
}
```

看下面的例子：

```
var example:String = "the actionScript cookbook";
// 显示: The ActionScript Cookbook
trace( StringUtilities.toTitleCase( example ) );
```

12.10. 修正空格符

问题

我想修正字符串首尾的空格符。

解决办法

使用自定义类方法*ascb.util.StringUtilities.trim()*，另外，如果你使用Flex 2 framework，则可以使用*mx.utils.StringUtil.trim()*静态方法。

讨论

字符串首尾的空格符总是让人很郁闷，一般我们都需要处理掉。ActionScript 并没有提供现成的*trim()* 实现，因此你必须自己实现了。

解决步骤如下：

把字符串分隔为字符数组。
删除开始部分的空格知道不是空格位置(tab, form feed, carriage return, newline, or space).
移除数组末尾的空格。
使用*join()*把字符数组连成新的字符串。

*ascb.util.StringUtilities.trim()*方法正式按照上面的步骤进行处理的，代码定义如下，这里还定义了一个判断字符是否为空格的方法*isWhitespace()*：

```
// 如果字符为空格则返回 true
public static function isWhitespace( ch:String ):Boolean {
   return ch == '\r' ||
               ch == '\n' ||
               ch == '\f' ||
               ch == '\t' ||
               ch == ' ';
}
public static function trim( original:String ):String {
   var characters:Array = original.split( "" );
   for ( var i:int = 0; i < characters.length; i++ ) {
      if ( isWhitespace( characters[i] ) ) {
         characters.splice( i, 1 );
         i--;
```

```
        } else {
            break;
        }
    }

    for ( i = characters.length - 1; i >= 0; i-- ) {
        if ( isWhitespace( characters[i] ) ) {
            characters.splice( i, 1 );
        } else {
            break;
        }
    }
    return characters.join("");
}
```

下面的例子演示了*trim()*方法的使用：

```
var example:String = "\n\r\f\ta string\t\t\n\n";
/*  调用trim( )方法之前显示如下：
this string value is:


        a string


<end>
*/
trace( "this string value is: " + example + "<end>" );
example = StringUtilities.trim( example );
//显示: this string value is: a string<end>
trace( "this string value is: " + example + "<end>" );
```

Flex 2 framework 实现了trim方法，入下面的例子：

```
//  显示: a string<end>
trace( StringUtil.trim( "\n    \r\ta string\t\t\n\n" ) + "<end>" );
```

12.11. 反转字符串

问题

我要把字符串进行反转处理

解决办法

使用*split()*方法创建字符数组，然后调用数组的*reverse()*和*join()*方法。

讨论

不管是单词还是字符都可以用同样的反转处理。唯一不同的是split()方法的分隔符。

处理步骤：

分隔字符串为数组，使用空格作为分隔符。
调用*reverse()*方法进行反转。
使用*join()*方法重新组织为字符串，当你反转单词时，使用空格作为连接符，当反转字符时使用空字符串作为连接符：

下面的代码演示了处理过程：

```
var example:String = "hello dear reader";
//分隔数组为单词数组.
var words:Array = example.split( " " );
// 反转数组
words.reverse(    );
// 使用空格连接数组元素为新的字符串
var exampleRevByWord:String = words.join( " " );
// 显示: reader dear hello
trace( exampleRevByWord );
// 分隔字符串为字符数组
var characters:Array = example.split( "" );
characters.reverse(    );
var exampleRevByChar:String = characters.join( "" );
// 显示: redaer raed olleh
trace( exampleRevByChar );
```

12.12. Unicode或ASCII 字符之间的转换

问题

我想得到字符的Unicode码或ASCII码。

解决办法

使用*String.charCodeAt()*和*String.fromCharCode()*方法

讨论

使用*fromCharCode()*显示的字符不能直接进入Flash文档，这个方法是静态的，这意味着要通过最顶层的*String*对象调用它。它接受一个整数或整数序列，然后转为等价的字符，当值小于128时*fromCharCode()* 实际上就是把ASCII码转为字符：

```
/*  显示:
    New paragraph: &#182;
    Cents: &#162;
    Name: Darron
*/
trace( "New paragraph: " + String.fromCharCode( 182 ) );
trace( "Cents: " + String.fromCharCode( 162 ) );
trace( "Name: " + String.fromCharCode( 68, 97, 114, 114, 111, 110 ) );
```

*charCodeAt()*方法则是把字符转为Unicode编码，如果编码小于128的话，*charCodeAt()*就是把字符转为等价的ASCII码：

```
var example:String = "abcd";
//第一个字符a  输出为97
trace( example.charCodeAt( 0 ) );
```

*fromCharCode()*方法还有个用处就是用Unicode 转义符来显示特殊字符：

```
var example:String = String.fromCharCode( 191 ) + "D"
                   + String.fromCharCode( 243 ) + "nde est"
                   + String.fromCharCode( 225 ) + " el ba"
                   + String.fromCharCode( 241 )   + "o?";

// 测试下第一个字符的编码是否是
// 191.，使用unicode 转义序列来代替fromCharCode( 191 ) ，产生特殊字符，如果是则显示：The string "&#191;D&#243;nde est&#225;
```

```
// el ba&#241;o?" has a &#191; at the beginning.
if ( example.charCodeAt( 0 ) == 191 ) {
   trace( "The string \"" + example + "\" has a \u00BF at the beginning." );
}
```

使用*charCodeAt( )*和*fromCharCode( )*方法可进行字符串的编码和解码处理。

下面的方法创建一个密码文字游戏，但是它仍然是不安全的，不能用于敏感型数据处理。

```
public static function encode( original:String ):String {
   // The codeMap property is assigned to the StringUtilities class when the encode(    )
   // method is first run. Therefore, if no codeMap is yet defined, it needs
   // to be created.
   if ( codeMap == null ) {
      // codeMap  属性是一个关联数组用于映射每个原始编码
      codeMap = new Object(    );
      //  创建编码为0到255的数组
      var originalMap:Array = new Array(    );
      for ( var i:int = 0; i < 256 ; i++ ) {
         originalMap.push( i );
      }
      //  创建一个临时数组用于复制原始编码数组
      var tempChars:Array = originalMap.concat(    );
      //  遍历所有原始编码字符
      for ( var i:int = 0; i < originalMap.length; i++ ) {
         var randomIndex:int = Math.floor( Math.random(    ) * ( tempChars.length - 1 ) );
         codeMap[ originalMap[i] ] = tempChars[ randomIndex ];
         tempChars.splice( randomIndex, 1 );
      }
   }
   var characters:Array = original.split("");
   for ( i = 0; i < characters.length; i++ ) {
      characters[i] = String.fromCharCode( codeMap[ characters[i].charCodeAt( 0 ) ] );
   }
```

```
}
public static function decode( encoded:String ):String {
    var characters:Array = encoded.split( "" );
    if ( reverseCodeMap == null ) {
        reverseCodeMap = new Object(    );
        for ( var key in codeMap ) {
            reverseCodesMap[ codeMap[key] ] = key;
        }
    }
    for ( var i:int = 0; i < characters.length; i++ ) {
        characters[i] = String.fromCharCode( reverseCodeMap[ characters[i].charCodeAt( 0 ) ] );
    }
    return characters.join( "" );
}
```

下面是具体使用方法：

```
var example:String = "Peter Piper picked a peck of pickled peppers.";
var encoded:String = StringUtilities.encode( example );
// 输出被编码的字符串，因为是随机产生的，每次输出都会不一样：
//
æT#Tm&#239;æ~*Tm&#239;*~NcT&#173;&#239;?&#239;*TNc&#239;?2&#239;*~Nc?T&#173;
&#239;*T**Tm:V
trace( encoded );
// 输出解码后的字符串
// 显示: Peter Piper picked a peck of pickled peppers.
trace( StringUtilities.decode( encoded ) );
```

14.0. 简介

日期和时间对于很多ActionScript程序来说是很重要的，比如用于一些和时间相关的定时操作，或者检测用户的登陆是否过期等。

在ActionScript内部是以毫秒的形式存储日期和时间的，但是很多编程语言的日期和时间是以秒为单位的，这点需要注意。

另外，*Date*类用于设置或获取日期和时间，或者直接通过其属性 fullYear, month 等，这些属性的值也是以毫秒为单位的。

14.1. 获得当前日期和时间

问题

我想知道当前日期和时间

解决办法

使用*Date()* 创建一个date对象，或者使用一个CGI脚本或其他服务端脚本返回服务器时间，然后根据返回值创建date对象

讨论

ActionScript计算出来的日期和时间是根据客户端计算机的日期和时间而得出的，因此如果客户端的时间不正确，那么date对象也是不正确的。

```
// 创建新的Date对象
var current:Date = new Date(    );
// 显示客户端日期和时间
trace(current);
```

如果你连接上了互联网，Flash可以从服务器获得日期和时间，这项技术可以保证获得正确的日期和时间(当然了你也可以故意把服务器时间设置错误，但是至少可以保证所有的客户端的时间是一致的)。

一般从服务器获得时间的步骤如下：

在Web服务器上创建CGI脚本，输出时间（秒）。
使用*flash.net.URLLoader*对象读取时间。
转换数值为number类型，乘以1000，再用这个数值重新构造一个date对象。

PHP是一种被广泛使用的脚本语言，用它输出服务器时间是很简单的：

```
<?php echo time(    );?>
```

如果你的服务器不支持PHP，或者说你更熟悉Perl (另一种脚本语言),输出也很简单：

```
#!/usr/local/bin/perl
print "Content-type:text/plain\n\n";
print time;
```

不管你使用什么服务端脚本，最后都要用ActionScript的*flash.net.URLLoader*对象载入服务器返回的时间值。

```
package {
   import flash.net.URLLoader;
   import flash.net.URLRequest;
   import flash.events.Event;
   public class ServerDateTimeExample {
      public function ServerDateTimeExample(    ) {
         // 下面的代码创建一个URLLoader对象，添加一个监听器，当数据载入
         // 完成时激活onDateTimeLoad( )方法
         var loader:URLLoader = new URLLoader(    );
         loader.addEventListener(Event.COMPLETE, onDateTimeLoad);
         loader.load(new URLRequest("script.cgi"));
      }
      private function onDateTimeLoad(event:Event):void {
         var loader:URLLoader = URLLoader(event.target);
         var data:int = parseInt(loader.data);
         var current:Date = new Date(data * 1000);
         trace(current);
      }
}
```

14.2. 获取时间值

问题

我要取得年月日，星期，小时，分，秒，毫秒等数值。

解决办法

是用*fullYear*, *date*, *month*, *day*, *hours*, *minutes*, *seconds*, *milliseconds* 等属性。

讨论

从Date对象的*fullYear*, *date*, *month*, *day*, *hours*, *minutes*, *seconds*, *milliseconds*属性中读取：

fullYear 属性返回4位数的年份值，如2010.
date 属性返回月天数，如1 到 31.
month 属性返回月份，如0 (1月) 到 11 (12月).
day 属性返回星期天数，如0 (星期天) 到 6 (星期六).
hours 属性返回小时数，如 0 到 23.
minutes 和 *seconds* 属性返回值为0 到 59.
milliseconds 属性返回值为 0 到 999.

14.3. 获取星期天数和月份名称

问题

我想知道星期几或几月

解决办法

创建包含星期天数和月份的名称数组，然后使用数字的天数和月份来提取相应的数组元素值。

讨论

ActionScript的*Date*类提供了day和month属性，它们返回整数值如星期（0到6），月份（0到11）。但是如果想获得名称而不是数字的话就需要自己创建包含星期天数和月份的名称数组，或者使用自定义类*ascb. util.DateFormat*。它已经定义了DAYS, DAYS_ABBREVIATED, MONTHS, 和 MONTHS_ABBREVIATED 等数组，细节如下：

```
public static const DAYS:Array = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];

public static const DAYSABBREVIATED:Array = ["Sun", "Mon", "Tues", "Wed", "Thurs", "Fri", "Sat"];

public static const MONTHS:Array = ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"];

public static const MONTHSABBREVIATED:Array = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"];
```

具体使用如下：

```
var example:Date = new Date(2010, 3, 10);

trace(DateFormat.DAYS[example.day]);   // 显示: Saturday

trace(DateFormat.DAYSABBREVIATED[example.day]);   // 显示: Sat

trace(DateFormat.MONTHS[example.month]);   // 显示: April

trace(DateFormat.MONTHSABBREVIATED[example.month]);   // 显示: Apr
```

14.4. 格式化日期和时间

问题

我要自定义日期和时间的显示格式

解决办法

使用*Date.toString()*, 或者自定义方法*DateFormat.format()*

讨论

*Date.toString()*方法返回Date对象的字符串类型数值，比如：

```
// 显示: Tue Jan 5 14:25:20 GMT-0800 2010
trace((new Date( )).toString( ));
```

如果忽略toString（）方法，ActionScript 也会自动调用*toString()*方法，结果是一样的，比如下面的例子：

```
// 也显示: Tue Jan 5 14:25:20 GMT-0800 2010
trace(new Date( ));
```

*Date*类本身并没有内建一些格式化日期和时间的方法，必须自己去实现了，当然创建Date对象时可通过一些字符串的组合实现，看下面的例子：

```
var example:Date = new Date(2010, 0, 5, 10, 25);
var formatted:String = (example.month + 1) + "/" + example.fullYear;
trace(formatted);  // 显示: 1/2010
```

这样的话，每次格式化日期和时间都要自己写一大段代码了，*ascb.util.DateFormat* 对象提供了一个 *format()* 方法实现了类似功能，*DateFormat*类是一个自定义类，专门设计来通过掩码格式化日期和时间。掩码可以是任意字符组合。

当创建*DateFormat*对象时，传递一个掩码字符串作为参数，下面的例子创建了一个*DateFormat*对象输出标准的美国日期格式：

```
var formatter:DateFormat = new DateFormat("m/d/Y");
```

一旦创建了*DateFormat*对象，调用format（）方法，通过指定的掩码格式化*Date*实例：

```
var example:Date = new Date(2010, 0, 5, 10, 25);
var formatter:DateFormat = new DateFormat("m/d/Y");
trace(formatter.format(example));  // 显示: 01/05/2010
```

可以通过mask属性的get和set方法操作mask字符串，也就是说可以改变已存在的*DateFormat* 对象的mask属性：

```
var example:Date = new Date(2010, 0, 5, 10, 25);
```

```
var formatter:DateFormat = new DateFormat("m/d/Y");
trace(formatter.format(example));  // 显示: 01/05/2010
formatter.mask = "m/d/Y h:i a";
trace(formatter.format(example));  // 显示: 01/05/2010 10:25 am
```

用单引号围绕的字符串可原样输出，这样就可以自定义格式化形式了：

```
var example:Date = new Date(2010, 0, 5, 10, 25);
var formatter:DateFormat = new DateFormat("m/d/Y at h:i a");
trace(formatter.format(example));  // 显示: 01/05/2010 am31 10:25 am
formatter.mask = "m/d/Y 'at' h:i a";
trace(formatter.format(example));  // 显示: 01/05/2010 at 10:25 am
```


14.5. 格式化秒或毫秒为分或秒

问题

我想把秒或毫秒格式化为分或秒。

解决办法

使用自定义类*ascb.util.DateFormat.formatSeconds()* 或 *ascb.util.DateFormat.formatMilliseconds()* 方法。

讨论

ActionScript里很多值都是以秒或毫秒形式存在的，例如声音长度单位是毫秒。但是在大多数情况下我们希望显示单位为分或秒，这里我们提供了*DateFormat*实现了上述方法。

ascb.utils.DateFormat 类有两个方法把数字转换为 *mm:ss* 格式。 *formatSeconds()* 方法转换秒，而 *formatMilliseconds()* 方法转换毫秒。

14.6.DMYHMSM和毫秒之间的转换

问题

我想在DMYHMSM格式(天，月，年，小时，分，秒，毫秒等形式的时间) 和毫秒之间自由转换 。

解决办法

使用*time*属性

讨论

我们已经习惯时间和日期是以年月日的形式，例如时间10:25 a.m., Tuesday, January 5, 2010 很容易理解，但是像ActionScript等一些语言是以毫秒为单位存储时间的，因此在显示给用户看之前需要做个转换。

当我们用ActionScript构造一个日期时，可以以DMYHMSM 形式构造：

```
//构造一个日期 10:25 AM, Tuesday, January 5, 2010
var example:Date = new Date(2010, 0, 5, 10, 25);
```

ActionScript自动把日期转换为毫秒值，要获取这个值需要调用*Date*对象的*time*属性：

```
// For Pacific Standard Time, displays: 1262715900000
trace(example.time);
```

14.7.使用 Timer（定时器）

问题

我想通过一定的间隔或一定的延时轮询某个方法

解决办法

使用*flash.util.Timer* 类

讨论

flash.util.Timer 类允许通过添加时间事件或延时来调用方法。通过*Timer*构造器创建实例对象，传递一个毫秒数字作为构造器参数作为间隔时间，下面的例子实例化一个*Timer*对象每隔1秒钟发出事件信号：

```
var timer:Timer = new Timer(1000);
```

一旦创建了*Timer*实例，下一步必须添加一个事件监听器来处理发出的事件，*Timer*对象发出一个*flash.event.TimerEvent* 事件，它是根据设置的间隔时间或延时时间定时发出。下面的代码定义了一个事件监听器，调用*onTimer()*方法作为处理函数：

```
timer.addEventListener(TimerEvent.TIMER, onTimer);
function onTimer(event:TimerEvent):void {
   trace("on timer");
}
```

Timer 对象不会自动开始，必须调用*start()*方法启动：

```
timer.start(    );
```

默认情况下只有调用*stop()*方法后才会停下来，不过另一种方法是传递给构造器第二个参数作为运行次数，默认值为0即无限次，下面的例子设定定时器运行5次：

```
var timer:Timer = new Timer(1000, 5);
```

下面的代码设定定时器延时5秒执行*deferredMethod()*方法：

```
var timer:Timer = new Timer(5000, 1);
timer.addEventListener(TimerEvent.TIMER, deferredMethod);
timer.start(    );
```

14.9. 字符串转换为日期

问题

我想通过字符串创建*Date*对象

解决办法

使用*DateFormat*对象的*parse()*方法

讨论

ActionScript并没有提供现成的方法把字符串转换为日期，自定义类*ascb.util.DateFormat*的parse()方法把传递进来的字符串参数转换为日期格式返回一个新的*Date*对象。

```
var formatter:DateFormat = new DateFormat("m/d/Y");
```

创建好*DateFormat* 实例，下一步就是调用*parse()*方法，传递一个字符串作为掩码，返回新的*Date*实例：

```
// 先是: Sat May 1 00:00:00 GMT-0700 2010 (timezone offset may vary)
trace(formatter.parse("05/01/2010"));
formatter.mask = "m/d/Y 'at' h:i a";
// 显示: Sat May 1 22:25:00 GMT-0700 2010 (timezone offset may vary)
trace(formatter.parse("05/01/2010 at 10:25 PM"));
```


15.0. 简介

在应用程序，游戏或Web站点上使用声音可以大大争强用户体验。在Flash IDE里通常是通过导入声音库，把声音放入时间线帧，关联到电影剪辑等等。本章内容覆盖ActionScript 3.0 使用Sound类及其相关类进行声音编程。

*Sound*类用于载入外部MP3文件，因为这文件并不是嵌入到swf内的，因此需要根据URL进行载入，这要遵循域安全约束（第三章）。

这一章涉及的类有：

Sound
SoundChannel
SoundLoaderContext
SoundMixer
SoundTransform

这些类都在*flash.media*包中，因此在写例子之前别忘了先引入 flash.media.Sound 包。

15.1. 创建Sound对象及载入声音

问题

我想载入声音到SWF应用程序上。

解决办法

创建一个*Sound*对象，载入外部的声音文件

讨论

创建一个*Sound*对象很容易，这跟创建其他类实例差不多，首先记得先引入*Sound*类库：

```
import flash.media.Sound;
Var _sound :Sound = new Sound(    );
```

接着我们需要一个声音文件，比如*song.mp3*，和swf文件放在服务器的同一个目录下。

要载入声音文件到*Sound*对象上，首先创建一个*URLRequest*对象（需要导入*flash.net.URLRequest*）：

```
soundFile = new URLRequest("song.mp3");
_sound.load(soundFile);
```

可以把上面两句简写成下面那样：

```
package {
    import flash.display.Sprite;
    import flash.media.Sound;
    import flash.net.URLRequest;
    public class LoadSoundExample extends Sprite {
        private var _sound:Sound;
        public function LoadSoundExample (    ) {
            _sound = new Sound(    );
            _sound.load(new URLRequest("song.mp3"));
        }
    }
}
```

这个类有个属性：_sound，将通过它播放声音。注意上面的代码并没有真正开始播放，这仅仅是做好准备工作，具体如何控制声音播放将在 第 15.2 章 介绍。

最简捷的写法就是直接把*URLRequest*传递给sound构造器：

```
public function LoadSoundExample(    ) {
```

```
    _sound = new Sound(new URLRequest("song.mp3"));
}
```

当传递URLRequest给构造器时，Sound会自动调用*load()*方法载入声音数据。当只播放一首声音时这种技巧很有效简洁。
否则当播放多首最好自己调用 *load()*方法，最好的例子就是音乐播放器，当一首音乐被选择，它的路径即被传递给 *load()*方法，准备播放。

15.2. 开始和停止播放声音

问题

如何开始播放或停止播放声音。

解决办法

使用*play()*方法播放声音，使用*close()*方法停止播放。

讨论

播放声音很简单，只要调用Sound对象的*play()*方法即可，如：

```
_sound = new Sound(new URLRequest("song.mp3"));
_sound.play(    );
```

很简单吧，另外*play()*方法有一些可选的参数，具体请看第15.1章 和 15.10章.。
*close()*方法即不是停止正在播放的声音，也不是停止声音流，当再次播放，用load()方法载入声音前需要调用*SoundChannel*类。

15.3. 给声音数据设置缓冲

问题

我想让声音播放的更流畅些

解决办法

通过*SoundLoaderContext*类设置缓冲时间

讨论

*Sound*类播放声音的方式是把整个声音数据载入完毕才进行播放，这对于小文件可能没什么问题，但如果是个大的声音文件，我们更希望边载入边播放，减少等待时间，更利于用户体验。

根据声音的编码和网络宽带，有时候声音播放的速度可能比下载的速度还要快，这种情况下声音播放就会暂停等待数据下载，为了更好的处理这个环节，我们可以设置一个数据缓冲区，当声音数据下载到一定数量时再进行播放，这样的话即使下载速度偶尔变慢也不会影响正常播放。

默认下*Sound*对象只创建1秒钟的缓冲，也就是说要想立即播放也需要等待1秒钟的缓冲，缓冲区的数据播放完后要想再次播放还要至少等1秒钟缓冲时间。

如果不能确定网络状况或该声音文件的编码是高比特位的（每秒需要更多的比特信息），这时需要提高缓冲区大小以使播放的更流畅。创建*SoundLoaderContext*对象，构造器参数为缓冲的大小，单位为毫秒，如下面的代码创建了5秒缓冲：

```
buffer = new SoundLoaderContext(5000);
```

下面两种写法都可：

```
var request:URLRequest = new URLRequest("song.mp3");
var buffer:SoundLoaderContext = new SoundLoaderContext(5000);
_sound = new Sound(request, buffer);
_sound.play(    );
```

或者：

```
var request:URLRequest = new URLRequest("song.mp3");
var buffer:SoundLoaderContext = new SoundLoaderContext(5000);
_sound = new Sound(    );
_sound.load(request, buffer);
_sound.play(    );
```

15.4. 声音的起始播放位置

问题

我不想从头播放声音而是从某个位置开始播放

解决办法

设置*play()*方法的参数

讨论

很多情况下我们并不希望从头开始播放声音，换句话说，我们要剪掉前面一部分后开始播放.*Sound*对象提供了这种能力让我们轻松做到这一点。

如果调用*play()*方法不指定参数，即从头开始播放，如果传递一个毫秒为单位的可选参数就会从指定的位置播放，如下面的例子在5.5秒开始播放：

```
_sound.play(5500);
```

利用这个特性我们可以做类似记录点的系统，让听众可以自由选择收听任何部分声音，看下面的例子演示：

```
package {
    import flash.display.Sprite;
    import flash.media.Sound;
    import flash.net.URLRequest;
    public class CuePoints extends Sprite {
        private var _sound:Sound;
        private var _cuePoints:Array;
        public function CuePoints(  ) {
            _cuePoints = [0, 10000, 30000, 68000, 120000];
            _sound = new Sound(new URLRequest("song.mp3"));
            // Play from the third cuepoint (30 seconds in)
            playCuePoint(2);
        }
        public function playCuePoint(index:int):void {
            _sound.play(_cuePoints[index]);
        }
    }
}
```

15.5. 循环播放

问题

我想多次播放声音或重复播放。

解决办法

在*play()*方法中设置循环值

讨论

默认情况下播放声音只是从头到尾播放一次即停止，很多时候我们希望可以播放多次或循环播放，把它作为游戏或网页的背景音乐。这里我们可设置循环让音乐厅起来好像是很长的音乐。

Play()方法的第2个参数即为循环次数，在设置第2个参数时必先设置第1个参数，第1个参数表示起始播放位置：

```
_sound.play(0, 3);
```

上面的代码意为从头播放3次。

循环参数最小值为0，即播放一次，如果传递0或负数，仍然是播放一次。

这样的话好像没有办法无限制播放，不过你可以传递一个非常的数，如 int.MAX_VALUE，这个数非常大，大概等于 2,147,483,647。即使是声音只有 1 秒钟也要播放 70 年，可以把当作无限使用。

15.6. 获得声音文件的大小

问题

我想知道一个mp3文件的大小及当前载入的大小。

解决办法

通过Sound对象的*bytesTotal*和*bytesLoaded*属性

讨论

当载入音频文件时，最好能让用户看到当前载入数据的进度。最后有一个可视化的进度条，就像Windows Media Player或QuickTime Player那样。

这一节我们利用Sound对象的两个属性做个灰色的进度条，*bytesTotal*和*bytesLoaded*.。*bytesTotal*指当前播放的mp3文件的总大小，*bytesLoaded*指已经下载的数据大小，两者相除即为声音下载的百分比。

下面的例子建立一个*enterFrame*监听器，每帧画一次白色矩形作为声音总大小，在计算出下载量画一个灰色的矩形。

```
package {
    import flash.display.Sprite;
    import flash.media.Sound;
    import flash.net.URLRequest;
    import flash.events.Event;
    public class ProgressBar extends Sprite {
        private var _sound:Sound;
        public function ProgressBar(    ) {
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
            _sound = new Sound(new URLRequest("song.mp3"));
            _sound.play(    );
        }
        public function onEnterFrame(event:Event):void
        {
            var barWidth:int = 200;
            var barHeight:int = 5;
            var loaded:int = _sound.bytesLoaded;
            var total:int = _sound.bytesTotal;
            if(total > 0) {
                // Draw a background bar
                graphics.clear(    );
                graphics.beginFill(0xFFFFFF);
                graphics.drawRect(10, 10, barWidth, barHeight);
                graphics.endFill(    );
                // The percent of the sound that has loaded
                var percent:Number = loaded / total;
                // Draw a bar that represents the percent of
                // the sound that has loaded
                graphics.beginFill(0xCCCCCC);
                graphics.drawRect(10, 10,
                                  barWidth * percent, barHeight);
```

```
                graphics.endFill(    );
            }
        }
    }
}
```

15.7. 读取声音文件的ID3标签数据

问题

我想访问正在播放的mp3文件的信息，比如歌曲名，艺术家，专辑，分类等等。

解决办法

读取Sound对象的id3属性

讨论

MP3 文件大多包含一些如songname, artist, album, genre, year等元数据，不过这些并不是都有，但大多数情况下都有*songname*和*artist*标签。

通过Sound对象的id3属性可获得这些数据。

这个属性其实是*flash.media.ID3Info*的类实例，它包含下列属性：

album
artist
comment
genre
songName
TRack
year

所以，想读取歌曲名，可这样写：

```
_sound.id3.songName
```

如果歌曲还没有下载到swf中，id3数据是不能够被访问的，那到底怎么知道id3数据已经下载了呢，我们可以通过监听Sound对象的*ID3*事件，看下面的例子：

```
package {
    import flash.display.Sprite;
    import flash.media.Sound;
    import flash.net.URLRequest;
```

```
    import flash.events.Event;
    import flash.text.TextField;
    public class ID3Reader extends Sprite {
        private var _sound:Sound;
        public function ID3Reader (    ) {
            _sound = new Sound(new URLRequest("song.mp3"));
            _sound.addEventListener(Event.ID3, onID3);
            _sound.play(    );
        }
        public function onID3(event:Event):void {
            // Create a text field and display it
            var id3Display:TextField = new TextField(    );
            addChild(id3Display);
            id3Display.x = 10;
            id3Display.y = 20;
            id3Display.width = 200;
            id3Display.height = 200;
            id3Display.background = true;
            id3Display.multiline = true;
            id3Display.wordWrap = true;
            // Add some info about the song to the text field
            id3Display.text += _sound.id3.songName + "\n";
            id3Display.text += _sound.id3.artist + "\n";
            id3Display.text += _sound.id3.album + "\n";
            id3Display.text += _sound.id3.year + "\n";                    }
        }
    }
}
```

15.8. 判定音乐是否播放完毕

问题

开始播放后，我想知道什么时候播放完毕

解决办法

监听*soundComplete*事件。

讨论

很多情况下我们需要知道什么时候音乐会播放完毕，例如，音乐播放器的播放列表，需要判定是否播放完毕以便播放下一首音乐。

这一节我们将介绍*flash.media*包中的*SoundChannel*类。当我们调用*Sound*对象的*play()*方法时，它会返回一个*SoundChannel*对象，因此每一首正在播放的音乐都会产生一个*SoundChannel*对象，这些声道合并并且最终输出。

当声音播放完毕时，对应的*SoundChannel*对象会发出*soundComplete*事件，就是*flash.events.Event.SOUND_COMPLETE*。

下面的例子建立一个简单的播放列表：

```
package {
    import flash.display.Sprite;
    import flash.media.Sound;
    import flash.net.URLRequest;
    import flash.events.Event;
    import flash.media.SoundChannel;
    public class PlayList extends Sprite {
        private var _sound:Sound;
        private var _channel:SoundChannel;
        private var _playList:Array;          // the list of songs
        private var _index:int = 0;           // the current song
        public function PlayList(  ) {
            // 创建列表，开始播放
            _playList = ["song1.mp3",
                         "song2.mp3",
                         "song3.mp3"];
            playNextSong(  );
```

```
        }
        private function playNextSong(    ):void
        {
            // If there are still songs in the playlist
            if(_index < _playList.length) {
                // Create a new Sound object, load and play it
                // _playList[_index] contains the name and path of
                // the next song
                _sound = new Sound(    );
                _sound.load(new URLRequest(_playList[_index]));
                _channel = _sound.play(    );
                // Add the listener to the channel
                _channel.addEventListener(Event.SOUND_COMPLETE,
                                                    onComplete);
                // Increase the counter
                _index++;
            }
        }
        public function onComplete(event:Event):void
        {
            playNextSong(    );
        }
    }
}
```

这里变量_index 起始值为 0，则_playList[index]正好等于"song.mp3"，这将是第一首播放的歌曲，接着_index 自增，当 *soundComplete* 事件触发时将会播放下一首歌，直到_index 大于_playList 的长度。

15.9. 跟踪音乐播放进度

问题

我想知道当前播放的歌曲已经播放到什么位置了

解决办法

使用*Sound.length*得到歌曲的总长度，*SoundChannel.position*得到当前的播放位置

讨论

第15.6章 讨论了如何添加一个进度条既显示音乐的播放进度，也显示音乐的下载进度，这一节就来创建播放进度条。

这一节涉及如何跟踪音乐的播放进度，要做到这一点，必须知道两件事：音乐的长度和当前的播放位置。这两个属性分别在两不同的类里。长度属性在sound类里，播放位置在*SoundChannel*类里，和缓冲条一样，这两个值相除就是播放进度百分比。

但这里有点比缓冲条复杂，问题就是length属性值，它是不确定的，直到声音被下载完才确定，也就说它只表示已经被下载的那部分数据的长度，举个例子说，如果一个10分钟的音乐已下载了10%，那么length报告歌曲长度为1分钟（length和position的单位都是毫秒）

还好，只通过简单的数学算法就可得出声音文件的真实长度，只要长度除以缓冲百分比即可算出实际长度。那上一个例子说，1除以1/10等于10，得到声音的长度为10分钟。

缓冲百分比就是*bytesLoaded*/*bytesTotal*，所以长度的计算公式就是：

```
length /= percentBuffered;
```

下面的例子显示两个进度条：

```
package {
    import flash.display.Sprite;
    import flash.media.Sound;
    import flash.media.SoundChannel;
    import flash.net.URLRequest;
    import flash.events.Event;
    public class ProgressBar2 extends Sprite {
        private var _sound:Sound;
        private var _channel:SoundChannel;
        public function ProgressBar2(  ) {
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
            _sound = new Sound(new URLRequest("song.mp3"));
```

```
        _channel = _sound.play(    );
}
public function onEnterFrame(event:Event):void
{
        var barWidth:int = 200;
        var barHeight:int = 5;
        var loaded:int = _sound.bytesLoaded;
        var total:int = _sound.bytesTotal;
        var length:int = _sound.length;
        var position:int = _channel.position;
        // Draw a background bar
        graphics.clear(    );
        graphics.beginFill(0xFFFFFF);
        graphics.drawRect(10, 10, barWidth, barHeight);
        graphics.endFill(    );
        if(total > 0) {
                // The percent of the sound that has loaded
                var percentBuffered:Number = loaded / total;
                // Draw a bar that represents the percent of
                // the sound that has loaded
                graphics.beginFill(0xCCCCCC);
                graphics.drawRect(10, 10,
                                        barWidth * percentBuffered,
                                        barHeight);
                graphics.endFill(    );
                // Correct the sound length calculation
                length /= percentBuffered;
                   // The percent of the sound that has played
                   var percentPlayed:Number = position / length;
                // Draw a bar that represents the percent of
                // the sound that has played
```

```
                graphics.beginFill(0x666666);
                graphics.drawRect(10, 10,
                                  barWidth * percentPlayed,
                                  barHeight);
                graphics.endFill(    );
            }
        }
    }
}
```

15.10. 暂停和重新播放声音

问题

我想暂停一下过一会儿再继续播放音乐

解决办法

利用*SoundChannel*的*position*属性即可做到

讨论

在第15.2章，我们讲过调用Sound对象的*close()*方法可以停止播放，但是这样也停止了声音流，要想重新播放，必须再次调用*load()*方法。

还好，*SoundChannel*类提供了一个*stop()*方法，它可以让音乐暂停而不影响声音流中断，要想重新播放,调用*play()*方法即可。

你会发现，当再次调用*play()*方法时，音乐会从头开始播放而不是从暂停的地方开始，这个时候就要用到SoundChannel类的position属性了，把它作为play()方法的第一个参数，看下面的代码演示：

```
package {
    import flash.display.Sprite;
    import flash.media.Sound;
    import flash.media.SoundChannel;
    import flash.net.URLRequest;
    import flash.events.Event;
```

```
import flash.display.Sprite;
import flash.events.MouseEvent;
public class PlayPause extends Sprite {
        private var _sound:Sound;
        private var _channel:SoundChannel;
        private var _playPauseButton:Sprite;
        private var _playing:Boolean = false;
        private var _position:int;
        public function PlayPause(    ) {
                // Create sound and start it
                _sound = new Sound(new URLRequest("song.mp3"));
                _channel = _sound.play(    );
                _playing = true;
                // A sprite to use as a Play/Pause button
                _playPauseButton = new Sprite(    );
                addChild(_playPauseButton);
                _playPauseButton.x = 10;
                _playPauseButton.y = 20;
                _playPauseButton.graphics.beginFill(0xcccccc);
                _playPauseButton.graphics.drawRect(0, 0, 20, 20);
                _playPauseButton.addEventListener(MouseEvent.MOUSE_UP,
                                                        onPlayPause);
        }
        public function onPlayPause(event:MouseEvent):void {
                // If playing, stop. Take note of position
                if(_playing) {
                        _position = _channel.position;
                        _channel.stop(    );
                }
                else {
                        // If not playing, re-start it at
```

```
                    // last known position
                    _channel = _sound.play(_position);
                }
                _playing = !_playing;
            }
        }
    }
```

上面的代码创建了一个按钮，当点击按钮时，如果正在播放，则暂停并记录位置，再次点击按钮时，继续从纪录的位置播放。

15.11. 获得声音的音量

问题

我想知道正在播放的声音的音量是多少

解决办法

通过*SoundChannel.leftPeak* and 和 *SoundChannel.rightPeak* 属性

讨论

任何声音，当在播放时产生强或弱的声波，我们称之为振幅，ActionScript 3.0 可获得一个立体声的左右声道的振幅，分别为*SoundChannel*的*leftPeak*和*rightPeak*属性。

它们的值范围是0.0到1.0，1.0表示最大的音量，具体的控制是通过*SoundTransorm*对象(看第15.14章).。

下面的例子创建两个矩形条用来显示两个声道的振幅：

```
package {
    import flash.display.Sprite;
    import flash.media.Sound;
    import flash.media.SoundChannel;
    import flash.net.URLRequest;
    import flash.events.Event;
    public class SoundLevels extends Sprite {
        private var _sound:Sound;
        private var _channel:SoundChannel;
```

```
        public function SoundLevels(    ) {
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
            _sound = new Sound(new URLRequest("song.mp3"));
            _channel = _sound.play(    );
        }
        public function onEnterFrame(event:Event):void
        {
            var leftLevel:Number = _channel.leftPeak * 100;
            var rightLevel:Number = _channel.rightPeak * 100;
            graphics.clear(    );
            graphics.beginFill(0xcccccc);
            graphics.drawRect(10, 10, leftLevel, 10);
            graphics.endFill(    );
            graphics.beginFill(0xcccccc);
            graphics.drawRect(10, 25, rightLevel, 10);
            graphics.endFill(    );
        }
    }
}
```

15.12. 停止播放所有的音乐

问题

我要停止所有当前正在播放的音乐

解决办法

使用*SoundMixer*的*stopAll()*方法

讨论

当开始播放一个音乐时，它会产生一个*SoundChannel*对象，在一个swf里可以播放多个音乐，每个音乐都对应一个SoundChannel对象，声音某些方面由Sound对象本身控制，有些则有SoundChannel对象控制，最后所有正在播放的声音进行合成输出到扬声器上。这个重要的工作由*SoundMixer*对象完成，它的属性和方法会影响所有正在播放的音乐。

其中一个方法就是*stopAll()*，它会停止所有正在播放的声音。

虽然声音可以大大增强诸如游戏，Web站点的用户体验，但是你也应该提供关闭音乐的功能（毕竟不是所有的人都喜爱听），SoundMixer类的静态方法*stopAll()*正好用的上，因为是静态的，所以可以直接调用，像这样：

```
public function stopSounds(event:Event):void {
    SoundMixer.stopAll(    );
}
```

15.13. 读取音乐的声谱

问题

我想显示出声音的波形图（声谱）

解决办法

使用*SoundMixer.computeSpectrum()*填充一个字节数组，读取这个数组得到具体数据。

讨论

访问声谱数据是ActionScript 3.0 新增的特性之一，在早期的Flash版本时期，开发者也很想获得这些数据，需要借助一些第三方工具，但是它们的实现过于复杂且效率低。现在这个功能已经内置于*SoundMixer*类之中，再加上新的*ByteArray*类，用很少的代码即可显示数据。

首先让我们看一下*ByteArray*这个类，它是ActionScript 3.0新增的专门用于处理二进制数据，经过优化，效率高，它位于*flash.utils*包中。表面上看起来和一般的数组差不多，但是它的方法处理数据比一般数组快的多。

要获得声谱数据，首先要创建一个空的*ByteArray*，像这样：

```
var spectrum:ByteArray = new ByteArray(    );
```

*ByteArray*再作为*SoundMixer.computeSpectrum()*方法的参数，这个方法获得声音的快照并计算出左右声道的波形，每个通道取256个值，范围在-1.0到1.0。再把数据存到*ByteArray*中。

数据是准备好了，我们该怎么使用呢，我们需要循环遍历*ByteArray* 512次，调用*getFloat()*方法，前面256个值代表左声道，后256为右声道。

下面的例子通过*BitmapData*对象和*setPixel32()*方法显示两个声道的波形：

```
package {
    import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.media.Sound;
    import flash.media.SoundChannel;
    import flash.net.URLRequest;
    import flash.utils.ByteArray;
    public class Spectrum extends Sprite {
        private var _sound:Sound;
        private var _channel:SoundChannel;
```

```
        private var _spectrumGraph:BitmapData;
        public function Spectrum(    ) {
            // Create bitmap for spectrum display
            _spectrumGraph = new BitmapData(256, 60,true,0x00000000);
            var bitmap:Bitmap = new Bitmap(_spectrumGraph);
            addChild(bitmap);
            bitmap.x = 10;
            bitmap.y = 10;
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
            _sound = new Sound(new URLRequest("song.mp3"));
            _channel = _sound.play(    );
        }
        public function onEnterFrame(event:Event):void
        {
            // Create the byte array and fill it with data
            var spectrum:ByteArray = new ByteArray(    );
            SoundMixer.computeSpectrum(spectrum);
            // Clear the bitmap
            _spectrumGraph.fillRect(_spectrumGraph.rect,0x00000000);
            // Create the left channel visualization
            for(var i:int=0;i<256;i++) {
                _spectrumGraph.setPixel32(i,20 + spectrum.readFloat(    ) * 20, 0xffffffff);
            }
            // Create the right channel visualization
            for(var i:int=0;i<256;i++) {
                _spectrumGraph.setPixel32(i,40 + spectrum.readFloat(    ) * 20,0xffffffff);
            }
        }
    }
}
```

15.14. 改变声音的音量和平衡度

问题

我想改变音量或左右声道平衡

解决办法

创建*SoundTransform*对象，改变音量及平衡度，把该对象赋值给*SoundChannel*对象的*soundTransform*属性。

讨论

在以前的版本中，可直接通过Sound对象改变音量及平衡度，现在这些被抽象出来形成*SoundTransform*类。

*SoundChannel*对象有个*soundTransform*属性，它是*SoundTransform*的类实例。要改变音量或平衡度，先创建*SoundTransform*对象，对其赋值，再赋给*SoundChannel*的*soundTransform*属性，下面的例子设置音量为50%：

```
var _sound:Sound = new Sound(new URLRequest("song.mp3"));
var channel:SoundChannel = _sound.play(    );
var transform:SoundTransform = new SoundTransform(    );
transform.volume = .5;
channel.soundTransform = transform;
```

音量的值范围为0.0到1.0 ，同样设置平衡度如下：

```
var channel:SoundChannel = _sound.play(    );
var transform:SoundTransform = new SoundTransform(    );
transform.pan = -1.0;
channel.soundTransform = transform;
```

值范围为-1.0到1.0 。

还可以把这些值直接传给*SoundTransform*的构造函数，如：

```
var channel:SoundChannel = _sound.play(    );
var transform:SoundTransform = new SoundTransform(.5, -1.0);
channel.soundTransform = transform;
```

15.15. 编写一个音乐程序

问题

我想写一个全功能的音乐程序，比如一个 MP3 播放器

解决办法

利用前面讲到的知识即可做到

讨论

这一节将整合前面的技术到一个程序上：

```
package {
    import flash.display.Sprite;
    import flash.display.Stage;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;
    import flash.text.TextField;
    import flash.events.Event;
    import flash.events.MouseEvent;
    import flash.events.TimerEvent;
    import flash.media.Sound;
    import flash.media.SoundChannel;
    import flash.media.SoundTransform;
    import flash.net.URLRequest;
    import flash.text.TextFormat;
    import flash.utils.Timer;
    public class CookBookPlayer extends Sprite {
        private var _channel:SoundChannel;
        private var _displayText:TextField;
        private var _sound:Sound;
        private var _panControl:PanControl;
        private var _playing:Boolean = false;
        private var _playPauseButton:Sprite;
        private var _position:int = 0;
```

```
private var _spectrumGraph:SpectrumGraph;
private var _volumeControl:VolumeControl;
public function CookBookPlayer(    ) {
        // Stage alignment
        stage.scaleMode = flash.display.StageScaleMode.NO_SCALE;
        stage.align = flash.display.StageAlign.TOP_LEFT;
        // Enter frame listener
        var timer:Timer = new Timer(20);
        timer.addEventListener(TimerEvent.TIMER, onTimer);
        timer.start(    );
        _playing = true;
        // Display a text field
        _displayText = new TextField(    );
        addChild(_displayText);
        _displayText.x = 10;
        _displayText.y = 17;
        _displayText.width = 256;
        _displayText.height = 14;
        // Create a sound object
        _sound = new Sound(new URLRequest(""http://www.rightactionscript.com
        samplefiles/sample.mp3""));
        _sound.addEventListener(Event.ID3, onID3);
        _channel = _sound.play(    );
        // Create a bitmap for spectrum display
        _spectrumGraph = new SpectrumGraph(    );
        _spectrumGraph.x = 10;
        _spectrumGraph.y = 33;
        addChild(_spectrumGraph);
        // Create the Play and Pause buttons
        _playPauseButton = new PlayButton(    );
        _playPauseButton.x = 10;
```

```
        _playPauseButton.y = 68;
        addChild(_playPauseButton);
        _playPauseButton.addEventListener(MouseEvent.MOUSE_UP, onPlayPause);
        // Create volume and pan controls
        _volumeControl = new VolumeControl(    );
        _volumeControl.x = 45;
        _volumeControl.y = 68;
        addChild(_volumeControl);
        _volumeControl.addEventListener(Event.CHANGE,onTransform);
        _panControl = new PanControl(    );
        _panControl.x = 164;
        _panControl.y = 68;
        addChild(_panControl);
        _panControl.addEventListener(Event.CHANGE, onTransform);
}
public function onTransform(event:Event):void
{
        // Get volume and pan data from controls
        // and apply to a new SoundTransform object
        _channel.soundTransform = new SoundTransform(_volumeControl.volume,
                                                _panControl.pan);
}
public function onPlayPause(event:MouseEvent):void
{
        // If playing, stop and record that position
        if(_playing) {
                _position = _channel.position;
                _channel.stop(    );
        }
        else {
                // Else, restart at the saved position
```

```
            _channel = _sound.play(_position);
        }
        _playing = !_playing;
    }
    public function onID3(event:Event):void {
        // Display selected id3 tags in the text field
        _displayText.text = _sound.id3.artist + " : " +
                                    _sound.id3.songName;
        _displayText.setTextFormat(
                            new TextFormat("_typewriter", 8, 0));
    }
    public function onTimer(event:TimerEvent):void {
        var barWidth:int = 256;
        var barHeight:int = 5;
        var loaded:int = _sound.bytesLoaded;
        var total:int = _sound.bytesTotal;
        var length:int = _sound.length;
        var position:int = _channel.position;
        // Draw a background bar
        graphics.clear(    );
        graphics.beginFill(0xFFFFFF);
        graphics.drawRect(10, 10, barWidth, barHeight);
        graphics.endFill(    );
        if(total > 0) {
            // The percent of the sound that has loaded
            var percentBuffered:Number = loaded / total;
            // Draw a bar that represents the percent of
            // the sound that has loaded
            graphics.beginFill(0xCCCCCC);
            graphics.drawRect(10, 10, barWidth * percentBuffered,barHeight);
            graphics.endFill(    );
```

```
                    // Correct the sound length calculation
                    length /= percentBuffered;
                    // The percent of the sound that has played
                    var percentPlayed:Number = position / length;
                    // Draw a bar that represents the percent of
                    // the sound that has played
                    graphics.beginFill(0x666666);
                    graphics.drawRect(10, 10,barWidth * percentPlayed,barHeight);
                    graphics.endFill(    );
                    _spectrumGraph.update(    );
                }
            }
        }
}
// "helper classes"
// (This is an outside package, but it's available to classes
// in the same file)
import flash.display.Bitmap;
import flash.display.BitmapData;
import flash.display.Sprite;
import flash.events.Event;
import flash.events.MouseEvent;
import flash.filters.DropShadowFilter;
import flash.geom.Rectangle;
import flash.media.SoundMixer;
import flash.utils.ByteArray;
class PlayButton extends Sprite {
        public function PlayButton(    ) {
            // Draw the Play/Pause graphic
            graphics.beginFill(0xcccccc);
            graphics.drawRoundRect(0, 0, 20, 16, 4, 4);
```

```
            graphics.endFill(    );
            graphics.beginFill(0x333333);
            graphics.moveTo(4, 4);
            graphics.lineTo(8, 8);
            graphics.lineTo(4, 12);
            graphics.lineTo(4, 4);
            graphics.drawRect(10, 4, 2, 8);
            graphics.drawRect(14, 4, 2, 8);
            graphics.endFill(    );
        }
}
class SpectrumGraph extends Sprite {
        private var _spectrumBMP:BitmapData;
        public function SpectrumGraph(    )
        {
            // Bitmap to draw spectrum data in
            _spectrumBMP = new BitmapData(256, 30,true, 0x00000000);
            var bitmap:Bitmap = new Bitmap(_spectrumBMP);
            bitmap.filters = [new DropShadowFilter(3, 45, 0, 1,3, 2, .3, 3)];
            addChild(bitmap);
        }
        public function update(    ):void
        {
            // Get spectrum data
            var spectrum:ByteArray = new ByteArray(    );
            SoundMixer.computeSpectrum(spectrum);
            // Draw to bitmap
            _spectrumBMP.fillRect(_spectrumBMP.rect, 0xff666666);
            _spectrumBMP.fillRect(new Rectangle(1, 1, 254, 28), 0x00000000);
            for(var i:int=0;i<256;i++) {
                _spectrumBMP.setPixel32(i,
```

```
                                        10 + spectrum.readFloat(    ) * 10,
                                        0xff000000);
            }
            for(var i:int=0;i<256;i++) {
                _spectrumBMP.setPixel32(i,
                                        20 + spectrum.readFloat(    ) * 10,
                                        0xff000000);
            }
        }
}
class VolumeControl extends Sprite {
        public var volume:Number = 1.0;
        public function VolumeControl(    )
        {
            addEventListener(MouseEvent.CLICK, onClick);
            draw(    );
        }
        public function onClick(event:MouseEvent):void
        {
            // When user clicks the bar, set the volume
            volume = event.localX / 100;
            draw(    );
            dispatchEvent(new Event(Event.CHANGE));
        }
        private function draw(    ):void {
            // Draw a bar and the current volume position
            graphics.beginFill(0xcccccc);
            graphics.drawRect(0, 0, 102, 16);
            graphics.endFill(    );
            graphics.beginFill(0x000000);
            graphics.drawRect(volume * 100, 0, 2, 16);
```

```
        }
}
class PanControl extends Sprite {
        public var pan:Number = 0;
        public function PanControl(    )
        {
                addEventListener(MouseEvent.CLICK, onClick);
                draw(    );
        }
        public function onClick(event:MouseEvent):void
        {
                // When the user clicks bar, set pan
                pan = event.localX / 50 - 1;
                draw(    );
                dispatchEvent(new Event(Event.CHANGE));
        }
        private function draw(    ):void {
                // Draw the bar and current pan position
                graphics.beginFill(0xcccccc);
                graphics.drawRect(0, 0, 102, 16);
                graphics.endFill(    );
                graphics.beginFill(0x000000);
                graphics.drawRect(50 + pan * 50, 0, 2, 16);
        }
}
```

16.0. 简介

Flash播放器是有能力回放视频的，虽然也可以把视频内容嵌入到swf文件中，但大部分Flash视频内容都是以.flv格式存储的，通过ActionScript在运行时载入到Flash播放器，这样swf文件更小，更利于管理视频内容。

Flash 视频载入有两种形式：渐进式下载和流下载。flv 流视频需要流服务器，比如 Flash Media Server。相反，渐进式下载不需要额外的软件，不过 ActionScript 处理这两种的方式是相同的，这一章将讨论如何播放 flv 视频。

16.1. 载入并播放视频

问题

我想渐进式下载视频并播放

问题

使用*NetStream*对象载入并播放视频，使用*Video*对象显示视频内容

讨论

ActionScript 3.0 需要多个类同时工作来载入和回放Flash视频。你必须使用*NetStream*对象载入视频并控制回放，但是*NetStream*类只关心如何读取数据，至于这些数据是什么内容并不知道，因此就需要*Video*对象，*Video*对象得到*NetStream*的数据并显示到屏幕上。

*NetStream*构造函数需要一个*NetConnection*对象作为参数，*NetConnection*对象关联将要播放的视频。当Flash视频来自于Flash Communication Server或Flash Media Server，*NetConnection*对象指向服务器，但是如果是下载渐进式视频内容则*NetConnection*对象的连接字符串为null。下面的代码构造一个*NetConnection*对象并初始化，用来下载渐进式视频，注意先引入*flash.net.NetConnection*类：

```
var videoConnection:NetConnection = new NetConnection(    );
videoConnection.connect(null);
```

一旦创建了*NetConnection*对象并调用了*connect()*方法，再构造*NetStream*对象，把*NetConnection*对象引用传给*NetStream*构造器作为参数。下面的代码构造了个*NetStream*对象(先引入imported *flash.net.NetStream*)：

```
var videoStream:NetStream = new NetStream(videoConnection);
```

创建好*NetStream*对象后，创建*Video*对象，并把两者关联起来。

*flash.media.Video*类是个可视化对象，因此可以添加到显示列表，下面的代码构造了一个*Video*对象并添加到显示列表：

```
var video:Video = new Video(    );
addChild(video);
```

接着调用*Video*对象的*attachNetStream()*方法，和*NetStream*对象关联起来：

```
video.attachNetStream(videoStream);
```

*NetStream*类定义了一个*play()*方法用于载入并开始播放Flash视频，参数可以是相对或绝对URL。下面的代码表示载入并播放名为example.flv的视频：

```
videoStream.play("example.flv");
```

如果*.flv*文件和swf文件在同一个域里，*play()*调用不会被Flash Player安全机制拒绝，但是如果在不同的域时就需要一个安全策略文件。

当缓冲足够时，视频会被自动播放，我们可控制缓冲和监视载入进度，这些将在第16.7章.讨论。

如果 flv 文件是嵌入进来的元数据，则需要处理元事件，这些将在第 16.4 章.讨论。 另外如果 flv 文件包含提示点，则需要处理提示点事件，这些将在第 16.8 章.讨论。

16.2. 控制视频声音

问题

我想控制视频声音的音量及平衡

解决办法

使用*NetStream*对象的soundTransform 属性

讨论

如果 Flash 视频有音轨，则声音部分会自动随着视频播放。如果想控制音量及平衡度，则需要访问 *NetStream* 对象的 soundTransform 属性，来获得一个 *SoundTransform* 对象的一个引用。更多细节请看第 15.14 章.

16.3. 读取回放时间

问题

我想读取当前视频的回放时间

解决办法

使用*NetStream*对象的time属性

讨论

*NetStream*类的time属性是个只读属性，单位为秒，下面的例子用文本框显示回放时间：

```
textField.text = videoStream.time + " seconds";
```

注意其值没有经过四舍五入，也就是说有时候得到数可能诸如5.235 ，如果需要整数，可使用*Math.round()*, *Math.floor()*, 或 *Math.ceil()*。
因 time 属性是只读的，我们不能改变它，要想控制回放，可通过第 16.5 章讲到的 *seek()*方法。

16.4. 获得视频长度

问题

我想知道视频长度是多少

解决办法

使用*onMetaData()*回调函数读取长度的元数据值

讨论

*NetStream*类没有定义一个属性指明视频长度，不过大多数情况，可以从flv文件本身获得长度值。flv文件可以包含元数据，大多数视频编码器都包含长度元数据，单位为秒，我们可通过ActionScript读取它。

当一个*NetStream*对象载入一个flv文件，它会自动调用*onMetaData()*回调方法，该回调模型不同于ActionScript 3.0 APIs 中的一般事件模型，一般情况下我们通过使用*addEventListener()*. 对一个事件进行监听，但是元数据事件必须为一个对象定义一个叫*onMetaData()*方法，然后把这个对象赋值给*NetStream*对象的client属性。该方法自动传递一个类型为Object的关联数组，其包含flv文件的元数据信息。下面的例子是用*trace()*显示视频的长度元数据：

```
var client:Object = new Object(    );
client.onMetaData = function(metadata:Object):void {
      trace(metadata.duration);
};
```

```
videoStream.client = client;
```

上面的例子代码实际上并不实用，我们更喜欢把函数引用赋值给*NetStream*对象的onMetaData属性：

```
package {
    import flash.display.TextField;
    import flash.media.Video;
    import flash.net.NetConnection;
    import flash.net.NetStream;
    import flash.events.NetStatusEvent;
    import flash.display.TextFieldAutoSize;
    import flash.display.Sprite;
    import flash.events.Event;
    public class Example extends Sprite {
        private var _stream:NetStream;
        private var _video:Video;
        private var _playbackTime:TextField;
        private var _duration:uint;
        public function Example(    ) {
            _video = new Video(160, 120);
            _playbackTime = new TextField(    );
            _playbackTime.autoSize = TextFieldAutoSize.LEFT;
            _playbackTime.y = 120;
            _playbackTime.text = "test";
            _duration = 0;
            var connection:NetConnection = new NetConnection(    );
            connection.connect(null);
            _stream = new NetStream(connection);
            _stream.play("video.flv");
            var client:Object = new Object(    );
            client.onMetaData = onMetaData;
            _stream.client = client;
            _video.attachNetStream(_stream);
```

```
            addChild(_video);
            addChild(_playbackTime);
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
        }
        private function onMetaData(data:Object):void {
            _duration = data.duration;
        }
        private function onEnterFrame(event:Event):void {
            if(_duration > 0 && _stream.time > 0) {
                _playbackTime.text = Math.round(_stream.time) + " / " + Math.round(_duration);
            }
        }
    }
}
```


16.5. 控制回放时间

问题

我想控制视频的回放时间

解决办法

使用*NetStream*的*seek()*方法

讨论

*NetStream*对象的time属性不是回放时间吗，直接修改值吗？这是不行的，因为time属性是只读的，因此需要使用*seek()*方法。*seek()*方法需要一个参数指定播放的位置，单位为秒，下面的例子从视频开始5秒后的位置播放：

```
videoStream.seek(5);
```

当在渐进式下载视频中使用*seek()*时，只能在已经下载的那部分视频使用，例如，60秒的视频下载了一半，只能在30秒内的内容进行定位。
当使用 *seek()*时，Flash 播放器根据指定的位置跳到视频最近的帧。视频的帧速率是不固定的，帧速越高，需要带宽越多。

16.6. 缩放视频

问题

我想改变视频的显示尺寸

解决办法

设置*Video*对象的width和height属性，如果要根据视频编码时的尺寸播放，可使用videoWidth和videoHeight值

讨论

视频尺寸是由*Video*对象的width和heigh属性决定的。当构造Video对象时可指定宽度和高度，例如，初始化视频尺寸为160 x 1 2 0：

```
var video:Video = new Video(160, 120);
```

通过width和height属性可以任意改变尺寸大小，如：

```
video.width = 320;
video.height = 240;
```

*Video*类还定义了两个只读属性，videoWidth和videoHeight，他们返回视频编码时的尺寸，可以用这两个属性设置width和height属性：

```
video.width = video.videoWidth;
video.height = video.videoHeight;
```

需要注意的是，flv还没下载之前这两个属性是不可用的，因此必须等两个属性可用时才可，我们可监听netStatus事件达到目的：

```
videoStream.addEventListener(NetStatusEvent.NET_STATUS, onStatus);
private function onStatus(event:NetStatusEvent):void {
    if(_video.videoWidth > 0 && _video.width != _video.videoWidth) {
        _video.width = _video.videoWidth;
        _video.height = _video.videoHeight;
    }
}
```

16.7. 管理和监视缓冲及下载进度

问题

我想管理视频缓冲及监视下载进度

解决办法

使用bufferTime属性设置缓冲区大小，使用bytesLoaded和bytesTotal来监视下载进度

讨论

默认下，视频只缓冲0.1秒，可通过NetStream的bufferTime属性进行设置：

```
videoStream.bufferTime = 10;//设置缓冲区大小为10秒
```

客户端的带宽不仅相同，我们该怎么设置缓冲区大小以增强用户体验呢。首先一步就是要检测用户带宽，一旦得到用户带宽，有两种选择：

根据不同的带宽准备多个不同比特率的视频，选择合适的视频给用户。

如果要让所有用户观看同一个视频，则需要根据带宽设置每个用户的缓冲大小，低带宽缓冲区设大一些。

要想显示视频缓冲进度，可使用bufferLength和bufferTime属性，bufferLength属性是只读的，返回已经缓冲的秒数，通过bufferLength/bufferTime返回缓冲进。

bufferLength 属性告诉我们已经有几秒钟数据进入缓冲区了，但是并没有告诉我们有多少数据卑下载了。我们可通过 bytesLoaded 和 bytesTotal 属性确定有多少数据下载了。

16.8. 监听提示点

问题

我想监听flv中的提示点

解决办法

使用*onCuePoint()*回调函数

讨论

FLV格式可嵌入提示点，提示点有各种用处,如：

字幕或提示说明

同步动画

记录视频回放状态

当使用诸如Flash Video Exporter，On2 Flix (http:// www.on2.com)，或Sorenson Squeeze (http:// www.sorensonmedia.com)等编码器时会插入一些提示点在flv文件中。当播放这些视频时，经过提示点时Flash播放器会收到提示信息，作为元数据，提示点不是使用标准的事件模型，而是使用了叫做*onCuePoint()*的回调函数。和*onMetaData()* 一样*onCuePoint()*方法定义为Object，然后赋值给*NetStream*对象的client属性。*onCuePoint()*方法接受一个参数，类型为object，属性有：

name

提示点的名称

time

提示点所在位置的时间

type

无论是"event"或"navigation," 都由编码时所选择的类型决定

parameters

key/value 对组成的关联数组

看下面的例子演示：

```
var client:Object = new Object(    );
client.onCuePoint = onCuePoint;
videoStream.client = client;
Then define the method appropriately:
private function onCuePoint(cuePoint:Object):void {
      trace(cuePoint.name + " " + cuePoint.time);
}
```


16.9. 给视频添加滤镜

问题

我想应用滤镜效果（模糊，色彩，置换等）到视频上。

解决办法

把滤镜数组赋值给*Video*对象的filters属性

讨论

Video 类继承了 *DisplayObject* 类的 filters 属性，意味着可以把滤镜效果应用到 *Video* 对象上。

16.10. 暂停和继续播放视频

问题

我想暂停或继续播放视频

解决办法

使用*NetStream*对象的*pause()*方法

讨论

*NetStream*类的*pause()*方法可以暂停和继续视频的回放。当没有参数时，即改变视频的暂停状态（如果在播放则暂停，如果暂停则继续播放）：

```
videoStream.pause(    );
```

如果参数为true，如果视频在播放则暂停，如果已经暂停则无任何效果：

```
videoStream.pause(true);
```

相反，如果参数值为false，如果视频已经暂停则继续播放，如果正在播放则无任何效果：

```
videoStream.pause(false);
```

继续播放不能使用*play()*方法，*play()*方法的作用只是载入并进行视频播放的初始化工作。

16.11. 停止播放视频

问题

我想停止视频的下载和播放

解决办法

使用*NetStream*类的*close()*方法

讨论

*NetStream*对象的*close()*方法可停止视频的播放，*pause()*方法只是暂停播放，flv数据还是在继续下载，如果要完全停止视频下载，必须使用*close()*方法，如下：

```
videoStream.close(    );
```

当调用*close()*方法后，Flash Player删除内存中的flv数据，要想播放需要重新下载一遍。但是浏览器可能已经缓存了flv数据，因此第二次播放可能会快一些，但是这可能导致不是最新的数据，如果要想每次都是播放最新的数据，可以在url上加个唯一字符串，如下面的代码：

```
videoStream.play("video.flv?uniqueIndex=" + (new Date(    )).getTime(    ));
```

如果因为数字版权问题不想视频数据被缓存，既不能使用渐进式下载视频，这种情况可使用流播放，利用诸如Flash Media Server，这些内容已经超出本书范围，暂不详述。

16.12. 擦洗视频

问题

我想擦洗视频的回放（快进或倒退播放）

解决办法

联合使用滑竿控制器与*seek()*方法

讨论

一般的方法是通过滑竿控制器来快进或倒退播放视频，首先要创建滑竿控制器，下面的例子代码演示滑竿控制器的编写：

```
package com.oreilly.as3cb.components {
    import flash.display.Sprite;
    import flash.net.NetStream;
    import flash.events.Event;
    import flash.events.MouseEvent;
    import flash.geom.Rectangle;
    public class VideoScrubber extends Sprite {
        private var _thumb:Sprite;
        private var _track:Sprite;
        private var _stream:NetStream;
        private var _duration:Number;
        private var _scrubbing:Boolean;
        public function VideoScrubber(stream:NetStream, duration:Number) {
            _stream = stream;
            _duration = duration;
            _track = new Sprite(    );
            _track.graphics.lineStyle(    );
            _track.graphics.drawRect(0, -2.5, 100, 5);
            addChild(_track);
            _thumb = new Sprite(    );
            _thumb.graphics.lineStyle(    );
            _thumb.graphics.beginFill(0xFFFFFF);
```

```
            _thumb.graphics.drawRect(-5, -5, 10, 10);
            _thumb.graphics.endFill(    );
            addChild(_thumb);
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
            _thumb.addEventListener(MouseEvent.MOUSE_DOWN, onMouseDown);
            _thumb.addEventListener(MouseEvent.MOUSE_UP, onMouseUp);
        }
        private function onMouseDown(event:MouseEvent):void {
            _scrubbing = true;
            var rectangle:Rectangle = new Rectangle(0, 0, _track.width, 0);
            _thumb.startDrag(false, rectangle);
        }
        private function onMouseUp(event:MouseEvent):void {
            _scrubbing = false;
            _thumb.stopDrag(    );
        }
        private function onEnterFrame(event:Event):void {
            if(_duration > 0) {
                if(_scrubbing) {
                    _stream.seek(_duration * _thumb.x / _track.width);
                }
                else {
                    _thumb.x = _stream.time / _duration * _track.width;
                }
            }
        }
    }
}
```

上面的代码中，构造函数接收两个参数：*NetStream* 对象和视频长度。构造函数创建滑竿控制器，当用户点击滑竿时，调用 *startDrag()*方法并设置_scrubbing 为 true。当_scrubbing 为 true 时，代码调用 *seek()*方法，根据滑竿的比例传递回放时间给该方法进行定位。

16.13. 清除视频显示

问题

我想视频显示的内容

解决办法

调用*Video*对象的*clear( )*方法

讨论

当关闭*NetStream*对象时，它不会自动清除video显示的内容。视频的最后一帧内容仍显示在*Video*对象上，需调用*clear( )*方法进行清除，看下面的例子：

```
video.clear(    );
```

要从显示列表上删除Video对象需调用*removeChild( )*：

```
removeChild(video);
```


16.14. 检测用户带宽

问题

我想检测用户网络带宽以便优化视频回放

解决办法

通过下载一个图片，根据下载时间计算出用户的网速

讨论

遗憾的是Flash播放器并没有内建带宽检测系统，要想测出用户带宽，需要通过Flash播放器下载一个文件如JPEG文件，通过下载的大小和所花的时间可以计算出平均下载速度，根据8个比特等于1个字节，1000个字节等于1个kilobyte（KB），转换公式为：

```
kilobits = bytes / 1000 * 8;
```

下面的例子代码演示如何测试带宽：

```
package com.oreilly.as3cb.util {
    import flash.events.EventDispatcher;
    import flash.net.URLLoader;
    import flash.net.URLRequest;
    import flash.events.Event;
    import flash.util.getTimer;
```

```
public class BandwidthTest extends EventDispatcher {
        private var _downloadCount:uint;
        private var _bandwidthTests:Array;
        private var _detectedBandwidth:Number;
        private var _startTime:uint;
        public function get detectedBandwidth(    ):Number {
                return _detectedBandwidth;
        }
        public function BandwidthTest(    ) {
                _downloadCount = 0;
                _bandwidthTests = new Array(    );
        }
        // Run the bandwidth test.
        public function test(    ):void {
                // Use a URLLoader to load the data.
                var loader:URLLoader = new URLLoader(    );
                // Use a URL with a unique query string to ensure the data is
                // loaded from the server and not from browser cache.
                var request:URLRequest = new URLRequest("bandwidthtestimage.jpg?unique="
                + (new Date(    )).getTime(    ));
                loader.load(request);
                loader.addEventListener(Event.OPEN, onStart);
                loader.addEventListener(Event.COMPLETE, onLoad);
        }
        // When the file starts to download get the current timer value.
        private function onStart(event:Event):void {
                _startTime = getTimer(    );
        }
        private function onLoad(event:Event):void {
                // The download time is the timer value when the file has downloaded
                // minus the timer value when the value started downloading. Then
```

```
        // divide by 1000 to convert from milliseconds to seconds.
        var downloadTime:Number = (getTimer(    ) - _startTime) / 1000;
        _downloadCount++;
        // Convert from bytes to kilobits.
        var kilobits:Number = event.target.bytesTotal / 1000 * 8;
        // Divide the kilobits by the download time.
        var kbps:Number = kilobits / downloadTime;
        // Add the test value to the array.
        _bandwidthTests.push(kbps);
        if(_downloadCount == 1) {
            // If it's only run one test then run the second.
            test(    );
        }
        else if(_downloadCount == 2) {
            // If it's run two tests then determine the margin between the
            // first two tests.
            // If the margin is small (in this example, less than 50 kbps)
            // then dispatch a complete event. If not run a test.
            if(Math.abs(_bandwidthTests[0] - _bandwidthTests[1]) < 50) {
                dispatchCompleteEvent(    );
            }
            else {
                test(    );
            }
        }
        else {
            // Following the third test dispatch a complete event.
            dispatchCompleteEvent(    );
        }
    }
    private function dispatchCompleteEvent(    ):void {
```

```
                // Determine the avarage bandwidth detection value.
                _detectedBandwidth = 0;
                var i:uint;
                for(i = 0; i < _bandwidthTests.length; i++) {
                        _detectedBandwidth += _bandwidthTests[i];
                }
                _detectedBandwidth /= _downloadCount;
                // Dispatch a complete event.
                dispatchEvent(new Event(Event.COMPLETE));
            }
        }
}
```

可以用上面的代码进行带宽测试，代码如下：

```
var bandwidthTester:BandwidthTest = new BandwidthTest(    );
bandwidthTester.addEventListener(Event.COMPLETE, onBandwidthTest);
bandwidthTester.test(    );
```

当complete事件触发时，访问detectedBandwidth属性得到带宽值

```
private function onBandwidthTest(event:Event):void {
   trace(event.target.detectedBandwidth);
}
```

17.0. 简介

电影剪辑在运行时，大多数数据都存储在内容中，一旦电影剪辑关闭，那么这些数据也同时从内存中清除掉，如果想存储数据或者让客户端的两个电影剪辑共享数据该怎么办呢，要想办法把数据存储在Flash播放器外面。

ActionScript中，*SharedObject*类实现了客户端机器数据的持久性存储。有两种类型的共享对象：本地和远程，这章集中讨论local shared objects (LSOs).

Local shared objects 很类似于浏览器中 cookies，LSOs 的功能也和 cookies 很类似，如存储用户登陆网站的用户名，这样不必每次登陆都要输入用户名了，不过 LSOs 的功能不仅限于此，它比 cookies 要强大的多，他们不必在客户端和服务器端进行传输，且都是以 ActionScript 数据类型形式存储。与此对应的服务端有 remote shared objects (RSOs)， LSOs 不需要在客户端和服务端额外的软件即可工作。

17.1. 创建，打开 Local Shared Object

问题

我想当访问swf过程中存储一些信息

解决办法

使用LSO.

讨论

如同这章的简介里描述的那样，Flash的LSOs 就如同Web浏览器中的cookies，它们被一些开发者称为“超级cookies”，因为LSOs可以存储大量数据，且存储和读取的都是原生的ActionScript数据类型。

LSOs 默认的空间大小100 KB， 用户可以通过Flash Player's Settings Manager控制LSOs的使用空间大小，来严格限制被使用的空间。存储在客户端的LSO文件是一种二进制文件，扩展名为*.sol*。同一个域的Flash电影通过*flash.net.SharedObject*类读写*.sol*文件。

当*.sol*文件创建后，被放置在Flash播放器对应的应用程序数据目录，以Windows为例，目录为*C:\Documents and Settings\[username] \Application Data\Macromedia\Flash Player\#SharedObjects\[random character directory name]*，在 Mac OS X 上，目录为*/Users/[username] /Library/Preferences/Macromedia/Flash Player/#SharedObject/ [random character directory name]*。随机字符目录命名是为了安全考虑。 一些恶意的swf文件可能试图猜测shared object（共享对象）的名称或目录，以便从文件系统中读取LSO，进而提升访问权限。因此必须把路径随机化，这样猜路径几乎是不太可能了。

LSOs 的静态方法*getLocal()* 用于创建或打开共享对象，它至少需要一个参数指明共享对象名称：

```
var example:SharedObject = SharedObject.getLocal( "example" );
```

*getLocal()*方法首先试图定位找到 LSO，如果找不到则根据这个名字创建新的，否则则返回*SharedObject* 实例。

17.2. 写入数据到共享对象上

问题

我想添加数据到LSO上.

解决办法

给共享对象的data对象添加属性值

讨论

共享对象（Shared objects）有个内建的属性data，data属性类型为object，因此可以添加任何信息上去：

```
// 存储username值给example共享对象
example.data.username = "Darron";
```

和早期版本的ActionScript不同，现在不能直接把属性值赋值给共享对象本身了，如果这样做了编译器会直接报错：

```
example.variable = "This will cause a compile-time error.";
```

另外，直接把值赋值给data属性也不正确：

```
example.data = "This will cause another compile-time error.";
```

正确的写法应该这样：

```
example.data.variable = "This is the correct way to store data.";
```

可以直接存储ActionScript原生数据类型：

```
example.data.exampleArray = new Array( "a", "b", "c" );
example.data.exampleDate = new Date();
example.data.exampleObject = { key1: "example", key2: -12, key3: null };
```

但需要注意的是，不能存储可视化对象（例如 MovieClips,Sprite,Buttons,TextFields）

17.3. 保存本地共享对象

问题

我想保存LSO到客户机上

解决办法

使用*SharedObject.flush()*方法

讨论

以下几种情况会导致Flash自动试图保存LSO数据到硬盘上：当Flash播放器关闭时，当共享对象被垃圾回收时，当调用*SharedObject.clear()*方法时。但是自动保存功能并不很实用，因为还有很多原因需要及时保存共享对象数据，因此我们可以显式调用*SharedObject.flush()*方法：

```
var flushResult:String = example.flush(    );
```

*flush()*方法有个可选的参数用于指定最小的硬盘空间，单位为字节，默认为0，指用最小的空间正好存储本地共享对象。

当*flush()*方法触发时，它试图把数据写到客户端上，调用结果有三种：

如果用户拒绝存储或Flash播放器因某种原因导致存储失败，该方法会抛出一个*Error*。
如果本地存储空间不够导致数据不能保存，该方法返回*SharedObjectFlushStatus.FLUSHED*。
如果用户没有分配足够的空间，该方法返回*SharedObjectFlushStatus.PENDING*。

三种情况中，当*flush()*方法返回*SharedObjectFlushStatus.PENDING*常量时，用户可以选择授权或拒绝保存数据，当用户做出选择后，netStatus事件被激活，需要定义一个事件处理函数，当事件处理函数被触发时，传递进一个类型为*flash.events.NetStatusEven*的事件，检查info.code 属性值判断用户是同意(*SharedObject.Flush.Success*)还是拒绝(*SharedObject.Flush.Failed*)

这里有个例子调用*flush()*保存数据，处理可能返回的结果：

```
var example:SharedObject = SharedObject.getLocal( "example" );
example.data.someData = "a value";
try {
   var flushResult:String = example.flush(    );
   // If the flush operation is pending, add an event handler for
   // netStatus to determine if the user grants or denies access.
   // Otherwise, just check the result.
   if ( flushResult == SharedObjectFlushStatus.PENDING ) {
      // Add an event handler for netStatus so we can check if the user
      // granted enough disk space to save the shared object. Invoke
```

```
        // the onStatus method when the netStatus event is raised.
        example.addEventListener( NetStatusEvent.NET_STATUS, onStatus );
    } else if ( flushResult == SharedObjectFlushStatus.FLUSHED ) {
        // Saved successfully. Place any code here that you want to
        // execute after the data was successfully saved.
    }
} catch ( e:Error ) {
    // This means the user has the local storage settings to 'Never.'
    // If it is important to save your data, you may want to alert the
    // user here. Also, if you want to make it easy for the user to change
    // his settings, you can open the local storage tab of the Player
    // Settings dialog box with the following code:
    // Security.showSettings( SecurityPanel.LOCAL_STORAGE );.
}
// Define the onStatus() function to handle the shared object's
// status event that is raised after the user makes a selection from
// the prompt that occurs when flush( ) returns "pending."
function onStatus( event:NetStatusEvent ):void {
    if ( event.info.code == "SharedObject.Flush.Success" ) {
        // If the event.info.code property is "SharedObject.Flush.Success",
        // it means the user granted access. Place any code here that
        // you want to execute when the user grants access.
    } else if ( event.info.code == "SharedObject.Flush.Failed" ) {
        // If the event.info.code property is "SharedObject.Flush.Failed", it
        // means the user denied access. Place any code here that you
        // want to execute when the user denies access.
    }
    // Remove the event listener now since we only needed to listen once
    example.removeEventListener( NetStatusEvent.NET_STATUS, onStatus );
};
```

如果确切知道存储数据的大小，可直接给*flush()*传参数：

```
// Request 500 KB of space for the shared object.
var flashResult:String = example.flush( 500 * 1024 );
```

17.4. 从共享对象中读取数据

问题

我想读取先前写入到LSO的数据

解决办法

读取共享对象的data属性中的值

讨论

在客户端读取这些内容很简单，这些持久性数据都保存在共享对象的data属性里，因此像下面的语句这样读就可以了：

```
// Read the value of exampleProperty from the shared object,
// example, and display it in the Output window.
trace( example.data.exampleProperty );
```

通过读写数据，我们可以判定用户是不是头一次访问swf文件：

```
// Create a shared object and store some data in it
var example:SharedObject = SharedObject.getLocal( "example" );
if ( example.data.previouslyViewed ) {
    // The user has already viewed the .swf file before, perhaps
    // we skip an introductory help screen here.
} else {
    // This is the first time the user is viewing the .swf file
    // because previouslyViewed has not yet been set to true.
    // Set previouslyViewed to true so that the next time this
    // code is run we know the user has been here before.
    example.data.previouslyViewed = true;
    example.flush(    );
}
```

17.5. 删除共享对象中保存的数据

问题

我想删除共享对象中的某个属性值或者干脆删除整个共享对象

解决办法

使用*delete*删除共享对象的data属性中的值，或使用*clear()*方法清除整个共享对象

讨论

删除共享对象中的数据是很简单的，但是要注意方法，在ActionScript中我们经常看到删除对象或数组只要赋值为null或undefined即可，但是对于共享对象这样做却不行：

```
// 试图删除共享对象example的someVariable 值
// 这语句编译通过，但是并不是我们所期望的
example.data.someVariable = null;
```

因为共享对象中null和undefined都是有效的值，因此上面的代码并没有删除someVariable属性，只不过设置为null而已，正确的方法是使用*delete*命令，如：

```
// Remove someVariable from the example shared object.
delete example.data.someVariable;
```

*clear()*方法删除整个共享对象，实际上就是删除硬盘中的.*sol*文件，看下面的代码：

```
// Create a shared object and store some data in it
var example:SharedObject = SharedObject.getLocal( "example" );
example.data.someData = "a value";
// Displays: a value
trace( example.data.someData );
// Remove the shared object from the disk
example.clear(    );
// Displays: undefined
trace( example.data.someData );
```

需要注意的地方是清除数据后，共享对象的引用仍然是有效的，这是还是可以重新添加数据进行保存。

17.6. 序列化自定义类

问题

我想把自定义类实例存储到LSO

解决办法

使用*flash.net.registerClassAlias()*方法保留类型信息并把类实例添加到共享对象的data属性上。

讨论

LSOs 使用特殊的二进制格式，Action Message Format (AMF)，当要在LSO中存储类实例时，实例会被编码为包含属性的普通的object。这样当重新从共享对象中读取实例时，已经不是原来的类实例了，因为已不能根据类型信息解码回来。

*flash.net*包中的*registerClassAlias()*方法就是为解决这个问题的，这个方法的使用是很简单的，在AS1.0和AS2.0中写法是*Object.registerClass()*，但是在AS3.0里已经被删除了，取而代之的是*flash.net.registerClassAlias()*。

*registerClassAlias()*方法需要两个参数，第一个参数表示类的别名，可以用任意字符串表示别名，比如modal包中有个Person类，别名可以是modal.Person，第二个参数类引用。

```
registerClassAlias( "somePackage.ExampleClass", ExampleClass );
```

这个代码的作用是把这个类的信息存进 LSO，当读取数据时，Flash 播放器就知道这个 object 到底是什么类。

下面的例子完整实现了类实例的保存，首先创建自定义类：

```
// Create a Person class in the model package
package model {
    public class Person {
        private var _firstName:String;
        private var _age:int;
        public function Person(firstName:String, age:int) {
            _firstName = firstName;
            _age = age;
        }
        public function toString(    ):String {
            return _firstName + " is " + _age + " years old";
        }
    }
```

```
}
```

接着，编写主类读取和写入数据

```
package {
    import flash.net.registerClassAlias;
    import flash.net.SharedObject;
    import model.Person;
    public class Example {
        public function Example(    ) {
            // Map "model.Person" to the Person class
            registerClassAlias( "model.Person", Person );
            // Create a shared object and store a Person instance in it
            var example:SharedObject = SharedObject.getLocal( "example" );
            // Test to see if the person instance has been saved already
            if ( example.data.person == undefined ) {
                trace( "first time, saving person instance" );
                var person:Person = new Person("Darron", 24);
                // Write the class instance to the local shared object
                example.data.person = person;
            } else {
                trace( "person instance already saved, using stored values" );
            }
            /* Every time this code is executed, the following is displayed:
            Darron is 24 years old
            */
            trace( example.data.person.toString(    ) );
}
```

这里需要注意的是*registerClassAlias()*必须在*SharedObject.getLocal()*方法之前调用才有效。否则的话共享对象会把Person解释为普通的object类型进行存储。

17.7. Flash程序之间共享数据

问题

我想要同一个域中的两个swf文件能访问同一个LSO。

解决办法

当创建或打开LSO时指定本地路径参数。

讨论

默认情况下，LSOs存储的名称是唯一的，这主要是为了防止名称冲突，例如，在Windows XP，如果电影剪辑名称为*movie.swf*，放在*http://www.person13.com/ascb*路径下，写入时LSO名称为example，则保存的路径为：

C:\Documents and Settings\[*user name*]\Application Data\Macromedia\Flash Player\#SharedObjects\[*random directory name*]\person13.com\ascb\movie.swf\example.sol

*.swf*文件的名称都包含在路径中，这样的话其他的swf即使在同一个域和路径下也不会冲突，因为各自都有自己相关的LSO，但是有时候，如果想让两个电影剪辑访问同一个LSO,这时，调用*getLocal()*需传入一个可选参数：本地路径，来打开或创建LSO。

本地路径参数（*getLocal()*第二个参数）为绝对或相对路径字符串，指定LSO的存储位置，例如：

```
var example:SharedObject = SharedObject.getLocal( "example", "/" );
```

如果*movie.swf*放在*http://www.person13.com/ascb*，根据上面的代码那么LSO就会保存在：

C:\Documents and Settings\[*user name*]\Application Data\Macromedia\Flash Player\#SharedObjects\[*random directory name*]\person13.com\example.sol

这个目录的不同点是缺少电影剪辑的信息，这样创建的LSO可以被同一个域的其他flash电影所共享访问：

```
var example:SharedObject = SharedObject.getLocal( "example", "/" );
```

正确理解绝对路径和相对路径是很重要的，这里举个例子，比如有两个swf文件：*movieOne.swf* 和 *movieTwo.swf*. 两个都放在同一个域(http:// *www.person13.com*) 但是在不同的路径下。*movieOne.swf* 在 http:// *www.person13.com/ascb/firstGroup*, *movieTwo.swf* 在 http:// *www.person13.com/ascb/secondGroup*，这样*movieOne.swf* 能创建和读取的本地路径为：

/

/ascb

/ascb/firstGroup

movieTwo.swf 能创建和读取的本地路径为：

/

/ascb

/ascb/secondGroup

因此，如果要创建两个swf都能访问的公共LSO，则*getLocal()* 方法的第二个参数必须是(*/* 或 */ascb*)

为了展示如何共享LSO，我们看一下下面的例子：

1. 创建新的swf文件，包含下列代码，命名为*movieA.swf.*

```
var count:SharedObject = SharedObject.getLocal( "count" );
// 第一次读取，默认值为0
if ( count.data.value == undefined ) {
   count.data.value = 0;
} else {
// 每次读取,value自动加1
   count.data.value++;
}
// 创建text field显示值
var output:TextField = new TextField(    );
output.text = "count value: " + count.data.value;
addChild( output );
```

2. 在电脑上建个新目录，名称为*LSOTest.*
3. 在*LSOTest*目录下创建两个子目录：*movieAPath* 和 *movieBPath.*
4. 拷贝*movieA.swf*文件到两个的子目录
5. 重名名*movieBPath*目录下的*movieA.swf* 为 *movieB.swf*
6. 多打开及关闭*movieA.swf*几次，每次打开，count值都会加1
7. 多打开及关闭*movieB.swf*几次，每次打开，count值都会加1，但是注意到初始值是从0开始，也就是说，*movieA.swf*和*movieB.swf* 使用各自的LSO，虽然LSO的名字相同，但是放在不同的路径里。
8. 要让两个swf使用同一个LSO，必须指定本地路径参数，像这样：
9. var count:SharedObject = SharedObject.getLocal("count", "/");
10. 这将导致两个swf在同一个路径下寻找LSO，因此他们使用了同一个文件.

17.8. 控制LSO的容量大小

问题

我想控制LSO的硬盘占用空间

解决办法

使用*Security.showSettings()*方法或者访问Web站点的Flash Player Settings Manager.

讨论

默认LSO的大小为100 KB。在.第17.3节 介绍了*flush()*方法来申请获得一定的硬盘空间大小，如果请求的空间超出共享对象空间的最大值，则会提示用户是否同意分配空间，通过*flash.system.Security.showSettings()*方法可设置LSO空间的最大值。下面的代码打开了本地存储容量的对话框界面。

```
Security.showSettings( SecurityPanel.LOCAL_STORAGE );
```


18.0. 简介

当我们讨论Flash之间的交互时，有两种情况需要考虑，第一，两个Flash都在同一个客户端上彼此之间需要通信，另一种情况是两个Flash在不同的电脑上，他们之间需要通信。

在Flash播放器6之前的版本里，同一台电脑上的两个Flash要通信只有一个办法（不使用服务端的功能），那就是使用*fscommand()*函数执行JavaScript方法，JavaScript充当中间桥梁。但是这种方法是很笨拙且不可靠（各浏览器及版本对JavaScript的支持程度不同），Flash播放器6提供了本地连接（Local Connection），意味着通过一个电脑上的所有Flash都可以通过发送广播信号和监听广播来相互通信了，本地连接的特性有：

使用相对比较简单
完全由ActionScript实现，从Flash播放器版本6开始都可用

如果是两个不同电脑上的Flash通信则需要服务器技术了，这里有三种方法：

远程LSO与Flash Media Server (Flash Communication Server) 的配合使用可实现两个Flash互相发送和接收数据
通过socket连接，通过服务器来中转客户端数据
如果不考虑实时性，可通过服务器的轮寻方法技术用一定的时间间隔发送和接收信息达到通讯的目的

这一章将讨论通过 *LocalConnection* 实现互通信

18.1. 创建本地连接

问题

我想让同一台电脑上的两个Flash进行通信

解决办法

使用*flash.net*包中的*LocalConnection*类来收发数据。在接收端使用*LocalConnection.connect()*方法监听信息，定义一个函数触发*LocalConnection.send()*发送数据，两个Flash必须指定相同通信信道。

讨论

当两个或多个在同一个电脑上运行着的 Flash 可通过 *flash.net.LocalConnection* 进行通信，而不必关心这些 Flash 都是从什么服务器上下载来的。

多个Flash要想进行通信必须做三件事：

1. 设置接收端Flash的监听处理函数
2. 告诉接收端收到数据后做什么
3. 从发送端Flash发送数据给接收端

接收端通过指定名称的本地连接监听信息。要创建通信信道，则首先创建本地连接对象，并使用*connect()*方法进行监听：

```
import flash.net.LocalConnection;

// 在接收端创建本地连接

var receiver:LocalConnection = new LocalConnection(    );

// 通知本地连接实例监听来自于 "_exampleChannel" 通道的信息

receiver.connect( "_exampleChannel" );
```

正如上面的代码例子那样，最好取一个以下划线开头的通信信道名称，这样Flash播放器不会加入域信息，如果不是以下划线开头，则通信信道名称字符串将被自动转换成*domain:connectionName*，而不管是本地域还是远程域。

所有通过本地连接的通信都被映射到接收端本地连接实例，默认下，接收端本地连接实例寻找实例中同名的发送函数，例如，如果发送端Flash发送一条消息寻找名称为*example()*的方法，接收端本地连接对象寻找已定义的*example()*方法。在上面代码基础上，再创建一个函数并关联到实例的example名称上。

```
receiver.example = function ( ) {

   output.text = "communication received";

};
```

但是这样的代码在ActionScript 3.0中却不行了，默认情况下，*LocalConnection*类不允许在运行时动态添加属性和方法，在这里有三种解决办法：

1. 创建动态的LocalConnection子类，使用这个实例作为所有接收端本地连接
2. 创建LocalConnection子类重写发送数据的方法
3. 重定向客户对象的请求

下面我们看一下每一种解决办法

要想在运行时修改类的方法，该类必须申明为*dynamic*。在ActionScript3.0中最简单的方法就是继承*LocalConnection*类并申明为*dynamic*类，使用这个类作为接收端：

```
package {
   import flash.net.LocalConnection;
   // 创建动态的Location Connection  类
   dynamic public class DynamicLocalConnection extends LocalConnection {
      //
   }
}
```

现在用这个*DynamicLocalConnection*代替*LocalConnection*类，在用以前的方法创建接收函数：

```
// 在接收端Flash创建本地连接对象
var receiver:DynamicLocalConnection = new DynamicLocalConnection(    );
// 通知本地连接实例监听来自"_exampleChannel"通道的信息
receiver.connect( "_exampleChannel" );
// 因为DynamicLocalConnection类是动态的，所以我们可以在实例上创建
//新的函数来处理本地连接信息
receiver.example = function(    ):void {
   trace( "communication received" );
};
```

第一种方法的优点是很灵活，你可以把这个类放到其他项目中继续适用，缺点也很明显，就是太灵活了而无法为其定义接口，这不符合面向对象设计原则。

第二中方法也是继承*LocalConnection*类，并且创建指定的方法用于处理信息，例如下面的类定义了一个example( )方法

```
package {
   import flash.net.LocalConnection;
   public class ExampleLocalConnection extends LocalConnection {
      public function ExampleLocalConnection(    ) {}
      public function example(    ):void {
         trace("communication received");
```

```
    }
  }
}
```

第三种方法使用*LocalConnection*的属性client来重定向请求，client对象必须是public的方法，看下面的例子：

```
package {
  import flash.net.LocalConnection;
  public class Example {
    private var _localConnection:LocalConnection;
    public function Example(  ) {
      _localConnection = new LocalConnection(  );
      _localConnection.connect( "_exampleChannel" );
      _localConnection.client = this;
    }
    public function example(  ):void {
      trace("communication received");
    }
  }
}
```

我们已经设置好了接收的swf，下一步编写发送的swf文件，发送端使用*LocalConnection.send()*方法发送信息。*send()*方法的第一个参数指定通道名称，第二个参数指定接收端被调用的方法名称。当一个flash接收到信息时，他将会触发这个方法。

下面的例子触发名为example的远程方法：

```
//通过"_exampleChannel" 通道发送信息
//触发接收端的example()方法
var sender:LocalConnection = new LocalConnection(  );
sender.send( "_exampleChannel", "example" );
```

当使用*LocalConnection*类时，通信是多对一的关系，也就是多个发送方，但接收方只有一个，如果接收端试图连接另一个接收端已经打开的通道，则*connect()*方法将抛出异常，应该用*try...catch* 来检测是否能成功连接：

```
var receiver:LocalConnection = new LocalConnection(  );
try {
  receiver.connect( "_exampleChannel" );
```

```
} catch( e:Error ) {
   // Error   不能在_exampleChannel 通道上建立连接，因为这个连接已经被使用了
}
```

要让 *LocalConnection* 能正常工作，发送端和接收端必须在同一个电脑上却都在运行才可，如果在同一台电脑不同的时间进行通信，则要使用 LSO。

18.2. 发送数据

问题

我想在几个flash之间传送数据

解决办法

通过*send()*方法的第三个参数传递数据

讨论

*send()*方法的第三个参数即传递的数据，前两个参数是必须的：通道名称，接收端方法名称。需要注意的是这个方法名称不能和*LocalConnection*类内部的属性和方法名称冲突，否则*send()*将调用失败，*send*, *connect*, *close*, *allowDomain*, *allowInsecureDomain*, *client*, and *domain*.这些名称不能作为参数使用。

在接收端应创建一个接收方法来接收发送过来的数据，下面的例子中，接收端创建*example()*.作为接收函数：

```
package {
   import flash.net.LocalConnection;
   public class ExampleReceiver {
      private var _receiver:LocalConnection;
      public function ExampleReceiver(    ) {
         // 实例化接收端本地连接，监听"_exampleChannel"通道
         _receiver = new LocalConnection(    );
         _receiver.connect( "_exampleChannel" );
         _receiver.client = this;
      }
      public function example( str:String, num:Number, bool:Boolean):void {
         trace( "The parameters are: " + str + "\n" + num + "\n" + bool );
```

```
        }
    }
}
```

发送端的代码如下：

```
// 发送三个参数给接收端
var sender:LocalConnection = new LocalConnection(    );
sender.send( "_exampleChannel", "example", "a string", 6.5, true);
```

发送数据的数据类型不限于基本数据类型，还可以是复合类型：*Object*, *Array*, *Date*, *TextFormat*, 和*XML*.

另外也可以是自定义数据类型，发送和读取自定义类对象的方式和LSO的方式差不多，更多细节请看第17.6章, 基本步骤如下：

1. 定义自定义类，在发送和接收Flash中引入
2. 使用*flash.net.registerClass()*注册自定义类对象，注意在所有的flash中要用相同的名称别名
3. 通过本地连接的Send a class instance over a local connection with the *send()* method.

下面的例子演示如何传递自定义类对象,首先定义自定义类Person：

```
package model {
    public class Person {
        private var _firstName:String;
        private var _age:int;
        public function Person(firstName:String, age:int) {
            _firstName = firstName;
            _age = age;
        }
        public function toString(    ):String {
            return _firstName + " is " + _age + " years old";
        }
    }
}
```

接收端代码：

```
import flash.net.registerClassAlias;
import model.Person;
registerClassAlias( "model.Person", model.Person );
```

```
package {
    import flash.net.LocalConnection;
    public class ExampleReceiver {
        private var _localConnection:LocalConnection;
        public function ExampleReceiver(    ) {
            _localConnection = new LocalConnection(    );
            _localConnection.connect("_exampleChannel");
            _localConnection.client = this;
        }
        public function example( person:Person ):void {
            trace( person.toString(    ) );
        }
    }
}
```

发送端代码：

```
package {
    import flash.net.registerClassAlias;
    import model.Person;
    public class ExampleSender {
        public function ExampleSender(    ) {
            registerClassAlias( "model.Person", Person );
            // Create a Person instance to send across
            var person:Person = new Person("Darron", 24);
            // Create the local connection and send a Person instance
            // to the receiving movie.
            var sender:LocalConnection = new LocalConnection(    );
            sender.send( "_exampleChannel", "example", person );
        }
    }
}
```

18.3. 基于本地连接通信的有效性验证

问题

我想知道发送的数据是否被接收端顺利收到

解决办法

设置接收端发送确认信息给发送端

讨论

要确认信息是否被收到，需要在接收端返回给发送端一条确认信息，为了验证信息是否被收到，可以在接收端发送一条确认消息给发送端。由于本地连接自身的限制，一个通道不能用于双向通信，因此需要各自建立一条通道互不干扰，具体步骤如下：

1. 接收端和发送端的代码请看第18.1章.
2. 在接收端通过新的通道"_exampleChannelReceipt "发送确认信息，使用*this.send( )*方法把信息发送给原始发送端。
3. 在发送端调用*connect( )*方法建立新通道"_exampleChannelReceipt"用于接收信息。.
4. 在发送端创建处理方法接受信息。

下面是发送端和接收端代码片断：

首先是接收端代码，使用*DynamicLocalConnection*类：

```
// 创建接受代码监听 "_exampleChannel" 通道
var receiver:DynamicLocalConnection = new DynamicLocalConnection( );
receiver.connect( "_exampleChannel" );
receiver.example = function( ):void {
   this.send( "_exampleChannelReceipt", "receipt" );
};
```

发送端代码：

```
var sender:DynamicLocalConnection = new DynamicLocalConnection(  );
seder.send( "_exampleChannel", "example" );
// 监听 "_exampleChannelReceipt" 通道
sender.connect( "_exampleChannelReceipt" );
// 定义receipt( )方法接收返回信息
sender.receipt = function(  ):void {
   output.text = "Receiver has delivered sent receipt";
};
```

在进一步改进代码，实际上接收端事先可以不知道返回信息的通道，可以让发送端告诉接收端通道名称是什么，这样做对于多个发送端和一个接收端时很有用，看下面的例子：

```
sender.send( "_exampleChannel", "example", "_exampleChannelReceipt1" );
```

发送端告诉接收端自己的接收通道是什么，接收端再根据这个通道名称发送确认信息给发送端:

```
receiver.example = function( receiptChannelName:String ):void {
    this.send( receiptChannelName, "receipt" );
}
```

这样做使程序更具有灵活性。

18.4. 接受其他域的连接请求

问题

我想让Flash接受来自其他域中Flash的本地连接请求

解决办法

在接收端使用*allowDomain()*方法

讨论

默认下，接收端只接受来自同一个域的本地连接请求，不过可以使用*allowDomain()*方法允许或禁止其他域的连接请求。

在ActionScript 3.0中，*LocalConnection*实例的*allowDomain()*方法需要被显式调用，方法接受一个或多个字符串参数，指明允许的远程域，下面的例子中，我们允许*darronschall.com*域中的flash发送消息到当前flash中：

```
var receiver:LocalConnection = new LocalConnection(    );
receiver.connect( "_exampleChannel" );
// 允许darronschall.com 发送数据到"_exampleChannel"通道
receiver.allowDomain( "darronschall.com" );
```

可以设置允许多个域：

```
receiver.allowDomain( "macromedia.com", "adobe.com", "google.com" );
```

还有两个特殊的字符串可使用，~ 表示所有的域都允许，*localhost* 表示只有本机的允许。

本地连接的*domain*属性可以确定所在的域，这个属性是只读的，如果要设置当前域的flash都可以通信，可把该属性值传给*allowDomain()*方法：

```
receiver.allowDomain( receiver.domain );
```

receiver.allowDomain("darronschall.com", receiver.domain);

上面的代码除了接收本地信息之外还允许接收来自*darronschall.com*的信息

还有个与*allowDomain()*类似的方法*allowInsecureDomain()*方法，大多数情况下这两种方法是一样，不同点在于HTTPS的使用上，当flash来自于HTTPS，其里面的本地连接实例不能和来自于HTTP的flash通信，除非使用*allowInsecureDomain()*，默认下HTTP 和 HTTPS 的flash是不能通信的，即使是在同一个域。

我们并不推荐使用*allowInsecureDomain()*，这样可能会带来不安全因素：

receiver.allowInsecureDomain("adobe.com");

19.0. 简介

很多情况下我们需要发送数据到flash或从flash中读取数值，比如：

1. 发送表单数据到服务端脚本以便存储到数据库；
2. 发送电子邮件数据给服务端发送电子邮件；
3. 从文本文件中读取数据；
4. 从服务器端读取数据，这些数据可能来自于数据库；
5. 发送数据给服务端处理并返回结果，比如说用户登陆界面

当从一个URL中读取数据时，Flash播放器会把数据转换成以下三种形式之一：文本形式，原始二进制数据，或 URL编码变量,URL-编码遵循以下规则：

1. 每个变量名用=号与其值相关联，=号两边没有空格
2. 变量值被Flash读取后都被转换为字符串，如 artist=Picasso.
3. 当有多个名称/值 对时，每一对都用&符号分开，如 artist=Picasso&type=painting.
4. 空格被替换成+符号，不是%20，比如 title=The+Old+Guitarist. (空格和%20 也可以用，但是用+ 兼容性更好
5. 除了数字和字母，如空格都被转换成十六进制的转义序列，例如，"L'Atelier" 被转换成 L%27Atelier (%27 就是单引号)，看下面的转义序列表

Table 19-1. Common URL-encoded escape sequences

<space>	%20 或 +	.	%2E
!	%21	/	%2F
"	%22	:	%3A
#	%23	;	%3B
$	%24	<	%3C
%	%25	=	%3D
&	%26	>	%3E
'	%27	?	%3F
(	%28	@	%40
)	%29	|	%7C
*	%2A	~	%7E
+	%2B	\	%5C
`	%2C	^	%5E
-	%2D	_	%5F

看下面的一个URL-编码格式的变量：

artist=Picasso&type=painting&title=Guernica&room=L%27Atelier

在ActionScript 3.0里发送和读取的数据都会被改变。*LoadVars*类已经被*flash.net*的*URLLoader*所代替。*URLLoader* 支持新的字符集映射，他们是*URLRequest*, *URLVariables*, 和 *URLStream* 。这些类提供了比以前更强大更灵活的处理能力。
有一些是没有改变的，比如读取的数据依然遵循 Flash 播放器标准安全沙漏体系，也就是说读取的数据文件或脚本必须和 flash 在同一个域内。

19.1. 从文本文件中读取数据

问题

我想读取外部的文本文件的内容到flash上。

解决办法

使用*URLLoader.load()*方法和DataFormat.VARIABLES 读取URL-编码数据

讨论

读取URL-编码数据时应该使用*URLLoader.load()*方法。

*load()*方法需要一个*URLRequest*实例作为参数，该参数指向文本文件的URL，这个URL即可以是相对路径也可以是绝对路径。另外*URLLoader* 需要进行配置把文本数据解释成URL-编码变量。设置*URLLoader*的dataFormat属性为DataFormat.VARIABLES常量：

```
import flash.net.*;
// 首先创建URLLoader对象
var example:URLLoader = new URLLoader(    );
// 进行设置
example.dataFormat = DataFormat.VARIABLES;
// 读取绝对路径的URL
example.load( new URLRequest( "http://www.darronschall.com/example.txt" ) );
// 读取相对路径的URL，文本文件和swf文件在同一个目录下
example.load( new URLRequest( "example.txt" ) );
```

假设文本文件的内容如下：

```
someText=testing&someNumber=123
```

一旦调用了*load()*方法，Flash播放器就会试图读取URL的数据填充*URLLoader*的data属性。读取完成后，Flash会试图解码这些内容并发出complete事件，指示数据已经读取完毕。这时候你可以添加处理函数来处理了。

如果读取失败，根据失败的原因*URLLoader*会发出不同类型的异常事件，因此除了监听完成事件，还要监听那些异常事件。

*load()*调用引发的异常有：

httpStatus

当试图读取数据，Flash播放器检测出错误的HTTP请求时发出

ioError

当遇到致命错误导致下载终止时发出

securityError

当试图读取安全沙漏允许以外的域 数据时发出

下面的例子演示监听各种事件：

```
package {
   import flash.events.*;
   import flash.net.*;
   import flash.util.trace;
   public class Example {
      public function Example(    ) {
         // Create the URLLoader instance to be able to load data
         var loader:URLLoader = new URLLoader(    );
         // Define the event handlers to listen for success and failure
         loader.addEventListener( IOErrorEvent.IO_ERROR, handleIOError );
         loader.addEventListener( HTTPStatusEvent.HTTP_STATUS, handleHttpStatus );
         loader.addEventListener( SecurityErrorEvent.SECURITY_ERROR,handleSecurityError );
         loader.addEventListener( Event.COMPLETE, handleComplete );
         // Configure the loader to load URL-encoded variables
         loader.dataFormat = DataFormat.VARIABLES;
         // Attempt to load some data
         loader.load( new URLRequest( "example.txt" ) );
      }
      function handleIOError( event:IOErrorEvent ):void {
         trace( "Load failed: IO error: " + event.text );
      }
```

```
        function handleHttpStatus( event:HTTPStatusEvent ):void {
            trace( "Load failed: HTTP Status = " + event.status );
        }
        function handleSecurityError( event:SecurityErrorEvent ):void {
            trace( "Load failed: Security Error: " + event.text );
        }
        function handleComplete( event:Event ):Void {
            trace( "The data has successfully loaded" );
        }
}
```

如果读取成功，Flash播放器把数据存在*URLLoader*实例的data属性上并发出complete事件。下面的例子演示处理complete事件：

```
function handleComplete( event:Event ):void {
    // 把event target 转换为URLLoader
    var loader:URLLoader = URLLoader( event.target );
    // 输出获得的数据
    trace( "someText = " + data.someText );
    trace( "someNumber = " + data.someNumber );
}
```

假设文本文件的内容如下：

```
someText=ActionScript+3.0+Cookbook&someNumber=3
```

通过 ***for...in*** 语句遍历所有读取的变量：

```
function handleComplete( event:Event ):void {
    var loader:URLLoader = URLLoader( event.target );
    // Use a for . . . in loop to loop over all of the variables that
    // were loaded
    for ( var property:String in loader.data ) {
        // property 就是data里的变量名
        trace( property + " = " + loader.data[property] );
    }
}
```

19.2. 从服务端脚本中读取变量

问题

我想从服务端脚本(ColdFusion, Perl, PHP, etc.)中读取变量.

解决办法

使用*URLLoader.load()*方法和DataFormat.VARIABLES 读取由服务端脚本产生的URL-编码数据

讨论

ActionScript读取服务端脚本数据和读取文本文件的操作是一样的，当这些数据是从服务端数据库或其他资源中产生时，脚本必须输出为URL-编码的数据才行，如果你采用perl脚本，输出为URL-编码，就不需要其他任何调整了，看下面的例子：

```
#!/usr/bin/perl

# In a more practical example this value would be retrieved

# from a database or other server-side resource.

$someText = "test";

# 定义Content-Type 头.

print "Content-Type: text/plain\n\n";

# Output the name-value pair.

print "someText=$someText";
```

ColdFusion：

```
<cfsetting enablecfoutputonly="yes">

<cfprocessingdirective suppresswhitespace="yes">

   <cfset someText = "test">

   <cfoutput>someText=#someText#</cfoutput>

</cfprocessingdirective>
```

PHP：

```
<?php

$someText = "test";

echo "someText=$someText";

?>
```

上面的每个脚本例子中都返回一个变量：someText，用下面的ActionScript代码读取这个变量：

```
package {

   import flash.events.*;
```

```
    import flash.net.*;
    import flash.util.trace;
    public class Example(    ) {
        public function Example(    ) {
            // 创建URLLoader实例
            var loader:URLLoader = new URLLoader(    );
            // 定义事件处理函数
            loader.addEventListener( Event.COMPLETE, handleComplete );
            // 设置读取的是 URL-编码 变量
            loader.dataFormat = DataFormat.VARIABLES;
            // 尝试读取数据
            loader.load( new URLRequest( "getSomeText.cfm" ) );
        }
        private function handleComplete( event:Event ):void {
            var loader:URLLoader = URLLoader( event.target );
            trace( "someText = " + loader.data.someText );
        }
    }
}
```

如果你事先不知道变量名，可以使用 ***for...in*** 语句遍历 data。

19.3. 读取文本块(包括HTML和XML)

问题

我想读取文本块，如HTML或XML.

解决办法

使用*URLLoader.load()*和DataFormat.TEXT

讨论

ActionScript 3.0 处理数据的方式已经和ActionScript 1.0 和2.0完全不同了，在以前的版本中，*LoadVars* 实例有两个不同的回调函数用于处理通过URL读取的数据。当处理读取的数据时触发*onLoad()*回调函数，当数据读取完成时触发*onData()*回调函数。

*flash.net.URLLoader*类和*LoadVars*类则不同，它只有一个complete事件和dataFormat属性用于决定如何解释这些下载的数据。dataFormat属性如果为DataFormat.TEXT，则会把数据解释为普通的文本内容，如果为DataFormat.VARIABLES则解释为URL-编码的变量,默认情况下，*URLLoader*把数据解释为文本。

假设从外部读取一个名为*example.html*的HTML数据，内容大致如下：

```
<b>Title:</b> ActionScript 3.0 Cookbook<br />
<b>Authors:</b> Joey Lott, Darron Schall, Keith Peters<br />
<b>Publisher URL:</b> <a href="http://www.oreilly.com">www.oreilly.com</a>
```

下面的例子将读取并显示这些内容：

```
package {
   import flash.display.*;
   import flash.text.*;
   import flash.events.*
   import flash.net.*;
   public class HTMLLoadingExample extends Sprite {
      private var _output:TextField;
      public function HTMLLoadingExample(    ) {
         initializeOutput(    );
         loadData(    );
      }
      private function initializeOutput(    ):void {
         _output = new TextField( );
```

```
            _output.width = stage.stageWidth;
            _output.height = stage.stageHeight;
            _output.html = true; // Enable HTML for the text field
            addChild( _output );
        }
        private function loadData(    ):void {
            var loader:URLLoader = new URLLoader(    );
            // Instruct the loader to read the file as plain text - This line is not
            // necessary because the dataFormat is DataFormat.TEXT by default.
            loader.dataFormat = DataFormat.TEXT;
            // Register an event handler for when the data is finished downloading
            loader.addEventListener( Event.COMPLETE, handleComplete );
            // Load the HTML text from the example.html file
            loader.load( new URLRequest( "example.html" ) );
        }
        private function handleComplete( event:Event ):void {
            var loader:URLLoader = URLLoader( event.target );
            // Assign the htmlText of the text field to the HTML text that was contained
            // in example.html. The data property of the URLLoader is the file contents.
            _output.htmlText = loader.data;
        }
    }
}
```


data 属性的数据类型是根据 dataFormat 属性的设置而决定的，如果设置为 DataFormat.TEXT，则 data 属性的值为 String 类型，如果设置 DataFormat.VARIABLES，则是 Object 类型，如果设置为 DataFormat.BINARY，则 data 的数据类型为 *flash.util.ByteArray*。

19.4. 检测读取进度

问题

我想知道数据读取的进度

解决办法

监听*URLLoader*的progress事件

讨论

*URLLoader*类有个progress事件，当正在下载数据时该事件就会触发。Flash播放器传递一个*flash.events.ProgressEvent*实例给事件处理函数，以便检查该事件的bytesLoaded和bytesTotal属性。bytesLoaded属性是指已经读取的数据量，bytesTotal属性是指总共要读取的数据量。

下面的代码演示如何监听progress事件：

```
package {
   import flash.display.*;
   import flash.text.*;
   import flash.events.*
   import flash.net.*;
   public class CheckProgressExample extends Sprite {
      private var _output:TextField;
      public function CheckProgressExample(  ) {
         initializeOutput(  );
         loadData(  );
      }
      private function initializeOutput(  ):void {
         _output = new TextField( );
         _output.width = stage.stageWidth;
         _output.height = stage.stageHeight;
         addChild( _output );
      }
      private function loadData(  ):void {
         var loader:URLLoader = new URLLoader();
         // Listen for the progress event to check download progress
         loader.addEventListener( ProgressEvent.PROGRESS, handleProgress );
```

```
            loader.load( new URLRequest( "example.txt" ) );
        }
        private function handleProgress( event:ProgressEvent ):void {
            // Calculate the percentage by multiplying the loaded-to-total
            // ratio by 100
            var percent:Number = Math.round( event.bytesLoaded / event.bytesTotal * 100 );
            _output.text = " Loaded: " + event.bytesLoaded + "\n"
                                    + "    Total: " + event.bytesTotal + "\n"
                                    + "Percent: " + percent;
        }
    }
}
```

需要注意的是 *URLLoader* 不能边下载边处理数据，如果想这样可以用 *URLStream*，这将在下一节讨论。

19.5. 边下载边访问数据

问题

我想边下载边访问数据

解决办法

使用 *flash.net.URLStream* 实例可在下在过程中立即读取二进制数据

讨论

第19.4章 讨论了如何检测数据下载进度，但是由于*URLLoader*类本身的限制，只能等数据全部下载完才能读取数据，要想边下载边读取可使用*URLStream*类代替之。

URLStream 可以边下载边以二进制形式读取数据， *URLLoader* 的 dataFormat 属性为 DataFormat.BINARY时和*URLStream*非常类似，它们有相同的事件，关键的区别是处理progress事件的方式不同而已。

在 *URLLoader* 中， progress 事件在检测多少已下载的数据或显示下载进度方面有用，而 *URLStream* 中的 progress 事件允许使用 bytesAvailable 属性和下列方法如 *readInt()*, *readByte()*, *readBoolean()*检查下载的数据。
当使用 *URLLoader* 不是很有效时或访问的数据都是二进制时最好使用 *URLStream*。

下面的例子代码使用 *URLStream* 读取一个*.txt* 文件，访问文件中的字节数据：

```
package {
   import flash.display.*;
   import flash.events.*
   import flash.net.*;
   public class CheckProgressExample extends Sprite {
      public function CheckProgressExample(    ) {
         var streamer:URLStream = new URLStream(    );
         // Listen for the progress event to act on data
         // as it is received
         streamer.addEventListener( ProgressEvent.PROGRESS, handleProgress );
         streamer.load( new URLRequest( "example.txt" ) );
      }
      private function handleProgress( event:ProgressEvent ):void {
         // Cast the target of the vent as a URLStream
         var streamer:URLStream = URLStream( event.target );
         //  遍历所有已读取得字节数据
         while ( streamer.bytesAvailable > 0 ) {
            // Read a byte value and output it to the console window
            trace( "Read byte: " + streamer.readByte(    ) );
         }
      }
   }
}
```

从上面的代码中可以看到，在进行读之前最好检测下 bytesAvailable 属性，如果读取的字节超出缓冲的结果则会抛出 *EOFError* 异常。

19.6. 发送数据给服务端脚本

问题

我想发送数据给服务端脚本

解决办法

创建一个包含数据的*URLRequest*实例，并用*flash.net.sendToURL()*方法传递给服务端脚本,如果想在新的浏览器窗口中打开可使用*flash.net.navigateToURL()*方法，如果想知道执行结果，可使用*URLLoader.load()*方法。

讨论

如果不用处理结果，最好是用*flash.net.sendToUrl()*方法发送数据，例如发送一个web表单数据，而服务端脚本处理后不用返回结果显示出来。但是*sendToURL()*方法也就不会返回数据是否收到的确认消息了，因此在实际应用中并不很实用。如果只是想显示一个信息如"Thank you for submitting the form,"确认信息，表示服务端已成功收到，可使用*URLLoader.load()*方法，3.12节 和19.7节.都讨论过这个方法。

通过*sendToUrl*传递给服务端的*URLRequest*实例包含传递的数据以及传输方式，如果data属性设置为*URLVariables*实例，则发送名称/值 对到服务器，也可以是*flash.util.ByteArray*，通过 HTTP POST 方式传递二进制数据到服务器，或者是*String* 类型的数据作为XML-RPC (看http://www.xmlrpc.com)请求发送到服务器。下面的代码演示：

```
function sendData(    ):void {
   // Create a request that sends data to the process.cfm page
   var request:URLRequest = new URLRequest( "process.cfm" );
   // Create some variables to send, someText and someNumber.
   var variables:URLVariables = new URLVariables(    );
   variables.someText = "Some text to send";
   variables.someNumber = 26.2;
   // Set the data to be sent to the variables, created earlier
   request.data = variables;
   // Send the data to the script for processing
   sendToURL( request );
}
```

传递给*URLRequest*的URL既可以是绝对地址也可以是相对地址，这可以通过Flash播放器安全沙漏进行管理，有关安全方面看第3.12节。

```
//  设置url属性为绝对地址
```

```
request.url = "http://www.darronschall.com/cgi-bin/submitVars.cgi";
// 设置为相对于swf 文件的地址
request.url = "cgi-bin/submitVars.cgi";
```

通过*sendToURL()*方法发送的数据被服务器处理后，服务器返回的信息会被Flash播放器忽略掉，要想发送数据且打开指定的浏览器可使用*navigateToURL()*方法，这个方法也*sendToURL()*方法几乎一样，只是多了个参数，如果设为_blank即会在新的窗口显示响应结果，还可设置为_self 或_parent 。

```
// 在新的窗口中发送数据
navigateToURL( request, "_blank" );
```

如果想接收服务器返回的结果，可使用*URLLoader.load()*方法(看3.12节 和19.7节).

默认情况下，使用*sendToURL()* 或*navigateToURL()*方法，数据都是通过HTTP POST方法传输的，*URLRequest*的method属性指定数据传输的方式，如URLRequestMethod.GET指 HTTP GET 和URLRequestMethod.POST指HTTP POST。

```
var request:URLRequest = new URLRequest( "cgi-bin/submit.cgi" );
// 创建发送的变量
var variables:URLVariables = new URLVariables(    );
variables.someText = "Post me!";
request.data = variables;
// 设置传输方式HTTP POST
request.method = URLRequestMethod.POST;
// Send the request and open the response in a new window
navigateToURL( request, "_blank" );
```

19.7. 发送变量并处理返回结果

问题

我想发送变量到服务器脚本，并处理返回的结果

解决办法

使用*URLLoader.load()*方法且设置*URLRequest*实例的data属性

讨论

当需要处理服务器返回的结果时，应该使用*URLLoader.load()*方法。比如一个flash的商店程序，它的商品分类都存在服务器数据库里，当用户点击一个分类，flash发送这个分类id给服务器脚本并返回这个分类的商品数据。

URLLoader.load() 发送数据给服务器脚本的方式和*sendToURL()*，*navigateToURL()* 是一样的，下面的完整例子发送数据到服务器脚本并返回URL-编码的值：

```
package {
   import flash.display.*;
   import flash.text.*;
   import flash.events.*
   import flash.net.*;
   public class SendAndLoadExample extends Sprite {
      private var _output:TextField;
      public function SendAndLoadExample(    ) {
         initializeOutput(    );
         sendData(    );
      }
      private function initializeOutput(    ):void {
         _output = new TextField();
         _output.width = stage.stageWidth;
         _output.height = stage.stageHeight;
         addChild( output );
      }
      private function sendData(    ):Void {
         // Create a URLRequest to contain the data to send
         // to process.cfm
```

```
            var request:URLRequest = new URLRequest( "process.cfm" );
            // Create name-value pairs to send to the server
            var variables:URLVariables = new URLVariables(    );
            variables.method = "getProductDetail"
            variables.productId = 2;
            request.data = variables;
            // Create a URLLoader to send the data and receive a
            // response
            var loader:URLLoader = new URLLoader(    );
            // Expect the script to return URL-encoded variables
            loader.dataFormat = DataFormat.VARIABLES;
            // Listen for the complete event to read the server response
            loader.addEventListener( Event.COMPLETE, handleComplete );
            // Send the data in the URLRequest off to the script
            loader.load( request );
        }
        private function handleComplete( event:Event ):void {
            var loader:URLLoader = URLLoader( event.target );
            // Expect the script to return name and description variables.
            // Display these values in a text field on the screen.
            _output.text = "Name: " + loader.data.name + "\n"
                                + "Description: " + loader.data.description;
        }
    }
}
```

20.0. 简介

XML 是一种结构化的描述数据形式，因其简单，灵活，尤其是在数据交换和可移植等优点现已成为事实上的工业标准。

我们在使用ActionScript过程中，XML是经常碰到的，第19章 介绍了如何发送和读取URL-编码的数据格式。这种格式传递简单数据还可以但是如果是复杂的数据或Unicode字符串，XML因其结构化优点就表现出来了。例如，如果从一个文本文件中读取数据转换为基本类型如string, URL-编码数据，如下面的那样，使用*URLLoader*对象读取：

```
myString=a+string+value
```

但是当你从外部数据源读取的数据并不是都能表现为URL-编码格式的字符串。例如向下面的那样，没对数据是用*分开的，而每对键值对都是用|分开的：

```
myObject=prop0|val0*prop1|val1*prop2|val2
```

当读取到这些数据后，还将使用*String.split()* 来分离这些元素，虽然这样做也能达到目的，但是如果使用XML就简单多了。例如，同样的数据用XML表示是这样的：

```
<myObject>
  <prop0>val0</prop0>
  <prop1>val1</prop1>
  <prop2>val2</prop2>
</myObject>
```

XML格式体现出比URL-编码格式更多的优点，如：

1. 当手动创建XML或程序创建的（ColdFusion，PHP等）很容易表现复杂数据。
2. 大多数服务端脚本都提供了内建的支持读写XML数据的功能。
3. XML已经是所有程序和平台传输和存储数据的通用标准。

当然，在Flash播放器中传输数据的方式并不局限于XML一种，我们将在第19章，21章，和24章讨论其他通讯方式。这一章将重点讨论XML，一个交换数据的工业标准，而且使用它并不需要额外的服务端软件（Flash Remoting和Sockets）。XML已经成为ActionScript重要组成部分。

ActionScript 3.0 新增了新的操作XML的语法，即 *ECMAScript for XML*，也叫*E4X*，提供一种比Document Object Model (DOM)更简单更容易访问XML的新方式。使用E4X，你会发现操作XML比以前更简单了，另外如果你是头一次操作XML，那么E4X也是很容易学习的。

这一章我们将涉及下列一些技术名词：

XML document

包含XML的文件，也指读取和发送XML的数据，XML文档的概念不要和*XMLDocument*类搞混。

XML包

一个XML包指从整个XML文档中取出的片断

XML 节点e

XML最基本的块，节点可以是元素，文本节点，属性等等

XML 元素

这个术语和"Tag"意义类似，更确切地说，元素包含tags。元素必须有开始和结束标签(<element></element>)或(<element />).

Root 节点

XML层级元素中最顶层的元素

Text 节点

包含文本的节点，一般都在元素里面

Attribute（属性）

元素的一部分，如<element name="value">，name="value"就是属性.

XML 声明

典型的申明如<?xml version="1.0" ?>.

XML 树

XML 数据的节点层级构成 XML 树

20.1. 理解XML结构（读写XML）

问题

我想知道如何读写XML

解决办法

XML是以层级和标签为基础的，如果你熟悉HTML，那学习XML应该会很容易

讨论

虽然读写XML并不是ActionScript所专有，不过懂得如何这项技能仍然是很有好处的，如果你还不熟悉XML，没关系，然我们一步步学习它。

XML是结构化数据的表现形式，这意味着要显示定义数据内容，例如，没有XML，你的数据时这样的：

```
Jerry,Carolyn,Laura
```

使用XML后：

```
<family>
  <father>Jerry</father>
  <mother>Carolyn</mother>
  <sister>Laura</sister>
</family>
```

可以看出XML表现的数据内容更丰富，关于XML还有其他一些注意点：

- XML 以大量的节点为基础，像上面的例子中<family>是一个节点，这些节点被称为**元素**，还有Jerry, Carolyn,和Laura 这些节点被称为 ***text节点***。
- 每个XML元素都有一个开始和结束标签，如上面的<family>为开始标签，结束标签为</family>。
- 元素可嵌套节点（元素或text节点），如上面的例子中<family>元素是根节点，他包含<father>, <mother>, 和<sister>元素，这些嵌套的节点被称为子节点，每个子节点又可以嵌套节点，不过这几个子节点都是text节点。

还有另一种节点，我们称之为**属性**，属性是特殊的节点附属于一个元素。如果你熟悉HTML的话，对属性的概念应该很熟悉了，比如<a>元素包含HRef属性，下面的例子用属性代替嵌套节点的写法：

```
<family father="Jerry" mother="Carolyn" sister="Laura" />
```

你可能会问什么时候该使用属性而不是嵌套节点呢？为什么要这样呢？这个往往是个人喜好问题，有时候使用属性使XML更加易懂些，一般如果值内容比较短时建议使用属性，如果数据比较大最好还是用嵌套节点比较好。

当然也可以两者一起使用，如下面的例子，<article> 元素包含title和author属性，而把嵌套的text节点用于显示文章主体：

```
<article title="XML: It's Not Just for Geeks" author="Samuel R.
Shimowitz">
My friends couldn't believe it when I started working with XML.
I became an outcast, confined to my dark office illuminated only
by the glow of my trusty CRT.
</article>
```

20.2. 创建XML对象

问题

我想创建一个XML对象用于存储数据

解决办法

使用下列方式之一创建XML对象：

创建XML对象并直接用XML进行赋值；
传递XML字符串给*XMZ*构造函数
创建一个空的XML对象并使用E4X填充数据
创建空的对象，从外部读取XML数据

讨论

在ActionScript很多地方都会用到XML对象，下面是最简单的方式创建XML对象：

```
var example:XML = <abc><a>eh</a><b>bee</b><c>see</c></abc>;
```

这是E4X表达式的例子，注意等号右边没有引号，ActionScript编译器知道右边的表达式就是XML。

上面的XML表达式是静态的，现在看一下如何创建动态的XML，在XML表达式中可以用{和}引入变量。例如要创建一个包含用户名和分数的XML发送到服务端，可以这样：

```
// 创建两个变量
var username:String = "Darron";
var score:int = 1000;
var example:XML = <gamescore>
```

```
                    <username>{username}</username>
                    <score>{score}</score>
                </gamescore>;
```

也可以先创建字符串，在传入XML构造器：

```
var str:String = "<gamescore><username>" + username + "</username>"
                + "<score>" + score + "</score></gamescore>";
var example:XML = new XML( str );
```

20.3. 添加XML元素

问题

我想构造一个XML对象，然后往内添加元素

解决办法

使用E4X语法创建子元素并添加到XML树中。另外用*insertChildBefore()*和*insertChildAfter()* 方法更容易控制元素的添加。

讨论

在日常工作中经常碰到往XML对象里添加新节点，然后把XML传递给其他应用程序。用E4X语法添加新节点是非常简单的，只要用操作符（.），跟一般的对象属性操作基本类似，看下面的例子：

```
// 创建一个XML实例
var example:XML = <example />;
// 创建新的节点
example.newElement = <newElement />;
/* 显示:
   <example>
      <newElement/>
   </example>
*/
trace( example );
```

上面的代码中，通过newElement属性来添加新的元素，属性名和内容不一定要相同，如下面的例子：

```
var example:XML = <example />;
example.emptyElement = "";
/*  显示:
   <example>
      <emptyElement/>
   </example>
*/
trace( example );
```

除了用(。)运算符外还可以用（[和]），如：

```
var example:XML = <example />;
var id:int = 10;
example[ "user" + id ] = "";
/*  显示:
   <example>
      <user10/>
   </example>
*/
trace( example );
```

两种写法基本上通用，不过还是有区别的，如下面的例子编译器就会报错：

```
example.some-element = ""; //编译错误
```

这种情况下只能用[]来设置属性名了：

```
example[ "some-element" ] = "";
```

还有个问题需要注意，这种方法增加的节点都是添加在XML树的尾部，要解决这个问题，可用*insertChildBefore()*和*insertChildAfter()* 方法来控制插入的位置。*insertChildBefore()* 方法在当前元素位置前插入新元素，而*insertChildAfter()* 在当前元素的后面插入，看下面的例子：

```
var example:XML = <example/>;
example.two = "";
example = example.insertChildBefore( example.two, <one /> );
example = example.insertChildAfter( example.two, <three /> );
/*  显示:
<example>
   <one/>
```

```
    <two/>
    <three/>
</example>
*/
trace( example );
```

要注意的是这两个方法不是修改原来的 XML，而是返回新的 XML 实例。

20.4. 添加文本节点

问题

我想添加文本节点

解决办法

即可使用E4X语法创建文本节点并插入到XML树中，也可用*appendChild()*, *prependChild()*, *insertChildAfter()*, 和*insertChildBefore()* 方法进行更多控制，灵活插入。

讨论

插入文本节点的方法和第20.3章讲的插入元素是一样的，都是用(。)操作符，例如：

```
// 创建XML实例
var example:XML = <example/>;
example.firstname = "Darron";
example.number = 24.9;
example.boolean = true;
example.abc = ["a", undefined, "b", "c", null, 7, false];
/* 显示:
<example>
    <firstname>Darron</firstname>
    <number>24.9</number>
    <boolean>true</boolean>
    <abc>a,,b,c,,7,false</abc>
</example>
```

```
*/
trace( example );
```

这个例子同时添加了文本和元素节点，=号右边的值即作为左右元素节点的子文本节点。

要向进行更多的控制可是使用 *appendChild( )*, *prependChild( )*, *insertChildBefore( )*, or *insertChildAfter( )*。

```
var example:XML = <example/>;
// 添加名为 two 元素以及值为2的文本子节点
example.appendChild( <two>2</two> );
// 在当前节点前面插入one 元素
example.prependChild( <one>"number 1"</one> );
// 在当前节点后面插入
example.insertChildAfter( example.one[0], 1.5 );
//在指定节点前插入
example.insertChildBefore( example.two[0], <part>1.75</part> );
/* 显示:
<example>
   <one>"number 1"</one>
   1.5
   <part>1.75</part>
   <two>2</two>
</example>
*/
trace( example );
```

上面的例子代码既加入了元素节点(<one>, <part>, <two>) 也加入了文本节点(1.5)。

20.5. 在XML元素中添加属性

问题

我想为XML元素增加属性

解决办法

使用E4X的 @ 操作符

讨论

使用E4X的 @ 操作符可为元素添加新的属性，如：

```
elementNode.@attributeName = "value";
```

在元素节点后面使用(.)操作符，再跟上@ 操作符，指定属性名称，=号右边即是属性值：

```
var example:XML = <example><someElement/></example>;
// 添加属性
example.someElement.@number = 12.1;
example.someElement.@string = "example";
example.someElement.@boolean = true;
example.someElement.@array = ["a", null, 7, undefined, "c"];
/* 显示:
<example>
  <someElement number="12.1" string="example" boolean="true"
  array="a,,7,,c"/>
</example>
*/
trace( example );
```

当使用这种语法时，属性名必须是合法的变量名称，也就是说必须是数字，字幕和下划线组成且不能以数字开头。但是有时如果属性名包含一些特殊符号，则不能用@操作符，必须加上[]操作符，例如：

```
example.someElement.@["bad-variable-name"] = "yes";
```

在[]里还可用表达式产生动态属性名，这是种很实用的技巧：

```
example.someElement.@["color" + num] = "red";
```

20.6. 读取XML树中的元素

问题

我想读取XML对象中的子元素

解决办法

使用 *elements()* 方法返回 *XMLList* 类型的所有元素，并用 *for each* 循环遍历

讨论

E4X提供了一个很方便的*elements()* 方法，该方法返回所有XML对象的子元素节点，再通过 *for each* 循环即可访问整个XML树结构：

```
var menu:XML = <menu>
                    <menuitem label="File">
                      <menuitem label="New"/>
                    </menuitem>
                    <menuitem label="Help">
                      <menuitem label="About"/>
                    </menuitem>
                    This is a text node
                  </menu>;
for each ( var element:XML in menu.elements(    ) ) {
  /*  显示:
  File
  Help
  */
  trace( element.@label );
}
```

从上面的例子中我们看到*elements()* 方法只返回下一级的子元素节点，这里面不包括文本节点和下一级节点，要向访问整个XML结构，还需进行递归循环处理：

```
var menu:XML = <menu>
                    <menuitem label="File">
                      <menuitem label="New"/>
                    </menuitem>
                    <menuitem label="Help">
```

```
                    <menuitem label="About"/>
                </menuitem>
                This is a text node
            </menu>;

/*  显示:
File
New
Help
About
*/
walk( menu );

// A recursive function that reaches every element in an XML tree
function walk( node:XML ):void {
    // Loop over all of the child elements of the node
    for each ( var element:XML in node.elements( ) ) {
        // Output the label attribute
        trace( element.@label );
        // Recursively walk the child element to reach its children
        walk( element );
    }
}
```

20.7. 通过名字查找元素节点

问题

我想通过节点名字来查找元素

解决办法

直接使用 E4X 的 . 加上属性名语法来查找元素

讨论

E4X 操作XML对象是非常简单的，比如每个元素节点，可直接访问元素名：

```
var fruit:XML = <fruit><name>Apple</name></fruit>;
// 显示: Apple
trace( fruit.name );
```

看，就是这么简单，用点操作符(.)即可，再看一下更复杂点的例子：

```
var author:XML = <author><name><firstName>Darron</firstName></name></author>;
// 显示: Darron
trace( author.name.firstName );
```

还有种简便的方法，即使用双点操作符(..)来跳过一级访问，如：

```
var author:XML = <author><name><firstName>Darron</firstName></name></author>;
// 显示: Darron
trace( author..firstName );
```

当有多个元素节点具有相同名称时，可通过索引值访问，这有点像数组，使用中括号，根据索引值访问指定的元素节点，例如：

```
var items:XML = <items>
                    <item>
                      <name>Apple</name>
                      <color>red</color>
                    </item>
                    <item>
                      <name>Orange</name>
                      <color>orange</color>
                    </item>
                  </items>;
```

```
// 显示: Apple
trace( items.item[0].name );
// 显示: Orange
trace( items.item[1].name );
```

items.item 返回两个元素的 *XMLList* 对象，第一个元素的索引值为0，第二个为1，用 *length()* 方法可得到找到的元素节点个数：

```
// 显示: 2
trace( items.item.length(    ) );
```

如果想访问特定名称的元素节点，但又不知道有多少个，可用 *for each* 循环遍历：

```
var items:XML = <items>
                        <item>
                           <name>Apple</name>
                           <color>red</color>
                        </item>
                        <item>
                           <name>Orange</name>
                           <color>orange</color>
                        </item>
                     </items>;
// 使用双点操作符
for each ( var name:XML in items..name ) {
   /* 显示:
   Apple
   Orange
   */
   trace( name );
}
```

也可用[]操作符来访问：

```
var nodeName:String = "color";
var fruit:XML = <fruit><color>red</color></fruit>;
// Displays: red
trace( fruit[nodeName] );
```

需要注意的是[]操作符不能和双点操作符一起用

```
trace( fruit..[nodeName] ); // 导致编译错误
```

20.8. 读取文本节点

问题

我想解析出文本节点及其值

解决办法

使用E4X语法，或者用*text()*方法返回元素的文本节点的*XMLList*对象，再用*toString()*方法把文本节点转换为字符串，也可通过*int()* 或 *Number()* 将其转换为其他类型。

讨论

第20.4节 讨论了如何创建文本节点，这一节将讨论如何读取文本节点的内容。比如下面的XML包：

```
<book>
   <title>ActionScript 3.0 Cookbook</title>
</book>
```

根节点为<book>，其包含子元素节点<title>，<title> 元素又包含文本节点，其值为 ActionScript 3.0 Cookbook。

可通过点操作符，根据节点名称找到元素，使用*toString()*方法得到文本节点的值：

```
var book:XML = <book>
                    <title>ActionScript 3.0 Cookbook</title>
                </book>;
// 用E4X 语法访问title元素
var title:String = book.title.toString(    );
// 显示: ActionScript 3.0 Cookbook
trace( title );
```

另外可通过*Boolean()*，*int()*等函数转换文本节点数据类型：

```
var example:XML = <example>
                        <bool>true</bool>
                        <integer>12</integer>
```

```
                    <number>.9</number>
                </example>;
// Convert a text node of "true" to boolean true
var bool:Boolean = Boolean( example.bool );
// Convert a text node of "12" to an integer
var integer:int = int( example.integer );
// Convert a text node of ".9" to a number
var number:Number = example.number;
/*  显示:
true
12
.9
*/
trace( bool );
trace( integer );
trace( number );
```

上面的代码有个小小的问题，即 *Boolean()* 转换可能会不成功，如果<bool> 元素的值如果是TRue则可能得不到正确的值，最好是进行一下大小写转换，如：

```
var bool:Boolean = example.bool.toLowerCase(    ) == "true";
```

再看一下面的例子，这个XML的根节点即包含元素节点也包含文本节点，如果想显示<fruit>的文本节点内容该怎么写呢？

```
var fruit:XML = <fruit>
                    <name>Apple</name>
                    An apple a day...
                </fruit>;
var value:String = fruit.toString(    );
/*  显示:
<fruit>
   <name>Apple</name>
   An apple a day...
</fruit>
*/
```

```
trace( value );
```

这种情况下，用*text()* 方法可正确返回文本节点内容了：

```
var fruit:XML = <fruit>
                    <name>Apple</name>
                    An apple a day...
                </fruit>;
for each ( var textNode:XML in fruit.text(  ) ) {
  // 显示: An apple a day...
  trace( textNode );
}
```

20.9. 读取元素的属性

问题

我想解析出元素的属性值

解决办法

使用 *attributes()* 方法返回指定元素的属性列表，或者通过名称用E4X的@操作符或*attribute()* 方法访问属性

讨论

通过*attributes()* 方法返回当前元素节点所有属性的*XMLList* 对象，*XMLList*对象是可索引的，很像*Array*对象，可通过索引值访问属性值：

```
var fruit:XML = <fruit name="Apple" color="red" />;
var attributes:XMLList = fruit.attributes(  );
// 显示: Apple
trace( attributes[0] );
// 显示: red
trace( attributes[1] );
```

上面的例子中只显示出属性值，用 *name()* 方法可显示出属性名，看下面的代码：

```
var fruit:XML = <fruit name="Apple" color="red" />;
// 显示: color
```

```
trace( fruit.attributes( )[1].name(    ) );
```

通过*for each* 循环语句可遍历所有属性名和属性值，如：

```
var fruit:XML = <fruit name="Apple" color="red" />;
for each ( var attribute:XML in fruit.attributes(    ) ) {
   /*  显示:
   name = Apple
   color = red
   */
   trace( attribute.name( ) + " = " + attribute.toString(    ) );
}
```

下面是E4X的写法：

```
var fruit:XML = <fruit name="Apple" color="red" />;
//  显示: red
trace( fruit.@color );
```

还有种方法，可使用*attribute( )* 方法并传入属性名作为参数，得到属性值：

```
var fruit:XML = <fruit name="Apple" color="red" />;
//  显示: red
trace( fruit.attribute("color") );
```

用(*)和@操作符可访问所有属性值，有些类似于*attributes( )*方法：

```
var fruit:XML = <fruit name="Apple" color="red" />;
//  显示: Apple
trace( fruit.@*[0] );
//  显示: red
trace( fruit.@*[1] );
//  显示: 2
trace( fruit.@*.length(    ) );
```

E4X 语法是很强大的，如果XML饱含有多个元素且有相同名称的属性，比如下面例子中的price属性，看用E4X如何统计出price：

```
// Create a fictitious shopping cart
var cart:XML = <cart>
                        <item price=".98">crayons</item>
                        <item price="3.29">pencils</item>
```

```
                        <group>
                            <item price=".48">blue pen</item>
                            <item price=".48">black pen</item>
                        </group>
                    </cart>;
// Create a total variable to represent represent the cart total
var total:Number = 0;
// Find every price attribute, and add its value to the running total
for each ( var price:XML in cart..@price ) {
    total += Number(price);
}
//  显示: 5.23
trace( total );
```

如果属性名含有特殊字符，可用[]进行访问：

```
var example:XML = <example bad-variable-name="yes" color12="blue" />;
var num:Number = 12;
//  显示: yes
trace( example.@["bad-variable-name"] );
//  显示: blue
trace( example.@["color" + num] );
```

20.10. 删除元素，文本节点和属性

问题

我想删除XML对象中的元素节点，文本节点或属性

解决办法

使用 delete 关键字

讨论

上面几节我们学习了如何添加元素，文本节点和属性到XML对象上。现在我们讨论如何删除这些节点，秘密就在于delete 关键字，看例子：

```
var example:XML = <example>
                        <fruit color="Red">Apple</fruit>
                        <vegetable color="Green">Broccoli</vegetable>
                        <dairy color="White">Milk</dairy>
                    </example>;
// 删除fruit元素的color属性
delete example.fruit.@color;
// 删除dairy元素
delete example.dairy;
// 删除vegetable元素的文本节点
delete example.vegetable.text( )[0];
/* 显示:
<example>
   <fruit>Apple</fruit>
   <vegetable color="Green"/>
</example>
*/
trace( example );
```

delete只能一次删除一个节点，如果要删除多个节点，可通过for循环进行删除：

```
var example:XML = <example>
                        <fruit color="red" name="Apple" />
                    </example>;
```

```
// Get an XMLList of the attributes for fruit
var attributes:XMLList = example.fruit.@*;
// Loop over the items backwards to delete every attribute.
// By removing items from the end of the array we avoid problems
// with the array indices changing while trying to loop over them.
for ( var i:int = attributes.length(    ) - 1; i >= 0; i-- ) {
      delete attributes[i];
}
/*  显示:
<example>
   <fruit/>
</example>
*/
trace( example );
```


20.11. 载入XML

问题

我想从XML文档中或服务端脚本产生的XML中读取XML数据

解决办法

使用*URLLoader.load()* 方法且设置dataFormat属性为DataFormat.TEXT读取数据，通过complete事件处理函数转换载入的数据为*XML*实例

讨论

ActionScript 3.0中发送和读取数据由新的*URLLoader*及其相关类完成，读取XML也没有什么特殊的地方。

读取XML文件的步骤如下：首先创建*URLLoader*实例以简单文本形式读取数据，其dataFormat 属性必须设置为DataFormat.Text，监听并添加complete事件处理函数，看下面的例子演示：

```
package {
   import flash.display.*;
   import flash.events.*;
```

```
    import flash.net.*;
    import flash.util.*;
    public class LoadXMLExample extends Sprite {
        public function LoadXMLExample(    ) {
            var loader:URLLoader = new URLLoader(    );
            loader.dataFormat = DataFormat.TEXT;
            loader.addEventListener( Event.COMPLETE, handleComplete );
            loader.load( new URLRequest( "example.xml" ) );
        }
        private function handleComplete( event:Event ):void {
            try {
                // Convert the downlaoded text into an XML instance
                var example:XML = new XML( event.target.data );
                // At this point, example is ready to be used with E4X
                trace( example );
            } catch ( e:TypeError ) {
                // If we get here, that means the downloaded text could
                // not be converted into an XML instance, probably because
                // it is not formatted correctly.
                trace( "Could not parse text into XML" );
                trace( e.message );
            }
        }
    }
}
```

上面的例子中之所以用 *try...catch* 块，是考虑到读取的数据有可能不是 XML 格式数据，*TypeError* 异常就是不能成功转换为 XML 实例时抛出的。

20.12. 从不同域中读取XML

问题

我想从其他域中读取XML数据

解决办法

设置*crossdomain.xml* 策略文件，允许可访问的远程域

讨论

看 第3.12章 关于如何使用*crossdomain.xml* 策略文件。

20.13. 发送 XML

问题

我想把XML数据发送给服务端脚本

解决办法

通过*URLRequest*实例把XML数据包装起来，用*flash.net.sendToURL( )* 发送数据并忽略服务器的响应，用*flash.net.navigateToURL( )* 发送数据并把服务器的响应显示在指定窗口，或者用*URLLoader.load( )* 发送数据并处理服务器响应。

讨论

XML一般被用来在应用程序之间传输数据，因此创建XML并不仅仅用于Flash内部，一般都是从其他源中读取XML数据或创建XML数据并发送给其他应用程序。

这一节将讨论如何从Flash端发送XML数据给其他应用程序，实际上很多时候都是这样的，比如一个Flash游戏发送用户名和比分到服务器，或者发送XML数据给服务端触发某个功能模块对XML数据进行处理，这种处理被称为远程过程调用(RPC)，如果规定都使用XML进行传递信息又被称为XML-RPC (看 http://www.xmlrpc.com)，所以你已经看到了，给服务器发送XML是很常用的。

如第20.11节和第19章 所讲的，ActionScript 3.0 发送和读取数据的方法都在*flash.net*包中。以前的版本中*XML* 类专门有*send( )*和*sendAndLoad( )*方法用于发送XML给服务器，现在在ActionScript 3.0中必须用*URLRequest*对象。19.6节 和19.7节 讨论了*URLReuest* 的基本用法。

下面让我们来看一个完整的例子，首先创建客户端ActionScript代码，然后选择一种服务端解决方案，如服务端脚本Perl, PHP, 或ColdFusion。

```
package {
   import flash.display.*;
```

```
import flash.text.*;
import flash.filters.*;
import flash.events.*;
import flash.net.*;
public class XMLSendLoadExample extends Sprite {
    private var _message:TextField;
    private var _username:TextField;
    private var _save:SimpleButton;
    public function XMLSendLoadExample(    ) {
        initializeDispaly(    );
    }
    private function initializeDispaly(    ):void {
        _message = new TextField(    );
        _message.autoSize = TextFieldAutoSize.LEFT;
        _message.x = 10;
        _message.y = 10;
        _message.text = "Enter a user name";
        _username = new TextField(    );
        _username.width = 100;
        _username.height = 18;
        _username.x = 10;
        _username.y = 30;
        _username.type = TextFieldType.INPUT;
        _username.border = true;
        _username.background = true;
        _save = new SimpleButton(    );
        _save.upState = createSaveButtonState( 0xFFCC33 );
        _save.overState = createSaveButtonState( 0xFFFFFF );
        _save.downState = createSaveButtonState( 0xCCCCCC );
        _save.hitTestState = save.upState;
        _save.x = 10;
```

```
    _save.y = 50;
    // When the save button is clicked, call the handleSave method
    _save.addEventListener( MouseEvent.CLICK, handleSave );
    addChild( _message );
    addChild( _username );
    addChild( _save );
}
// Creates a button state with a specific background color
private function createSaveButtonState( color:uint ):Sprite {
    var state:Sprite = new Sprite(    );
    var label:TextField = new TextField(    );
    label.text = "Save";
    label.x = 2;
    label.height = 18;
    label.width = 30;
    var background:Shape = new Shape(    );
    background.graphics.beginFill( color );
    background.graphics.lineStyle( 1, 0x000000 );
    background.graphics.drawRoundRect( 0, 0, 32, 18, 9 );
    background.filters = [ new DropShadowFilter( 1 ) ];
    state.addChild( background );
    state.addChild( label );
    return state;
}
private function handleSave( event:MouseEvent ):void {
    // Generate a random score to save with the username
    var score:int = Math.floor( Math.random(    ) * 10 );
    // Create a new XML instance containing the data to be saved
    var dataToSave:XML = <gamescore>
                            <username>{username.text}</username>
                            <score>{score}</score>
```

```
                                    </gamescore>;
    // Point the request to the script that will handle the XML
    var request:URLRequest = new URLRequest( "/gamescores.cfm" );
    // Set the data property to the dataToSave XML instance to send the XML
    // data to the server
    request.data = dataToSave;
    // Set the contentType to signal XML data being sent
    request.contentType = "text/xml";
    // Use the post method to send the data
    request.method = URLRequestMethod.POST;
    // Create a URLLoader to handle sending and loading of the XML data
    var loader:URLLoader = new URLLoader(    );
    // When the server response is finished downloading, invoke handleResponse
    loader.addEventListener( Event.COMPLETE, handleResponse );
    // Finally, send off the XML data to the URL
    loader.load( request );
}
private function handleResponse( event:Event ):void {
    try {
        // Attempt to convert the server's response into XML
        var success:XML = new XML( event.target.data );
        // Inspect the value of the success element node
        if ( success.toString(    ) == "1" ) {
            _message.text = "Saved successfully.";
        } else {
            _message.text = "Error encountered while saving.";
        }
    } catch ( e:TypeError ) {
        // Display an error message since the server response was not understood
        _message.text = "Could not parse XML response from server.";
    }
```

```
    }
  }
}
```

URLRequest 对象的 contentType 属性默认设置为 application/x-www-form-urlencoded ，当发送 XML 数据时必须设置为 text/xml ，而且设置 method 属性为 *URLRequestMethod.POST* ，表示通过 HTTP POST.发送数据。

下一步就是创建服务端脚本，首先看一下Perl脚本，将下列代码保存为*gamescores.cgi* (或 *gamescores.pl*)：

```
#!/usr/bin/perl
# Flash/Perl+CGI XML interaction demo
# Arun Bhalla (arun@groogroo.com)
use strict;
use XML::Simple;
use CGI;
my $ScoreFile = "scores.txt";
# Here we assume that this CGI script is receiving XML in text/xml
# form via POST. Because of this, the XML appears to the script
# via STDIN.
my $input = XMLin(join('',<STDIN>));
# Write out the HTTP header
print CGI::header('text/xml');
# Try to open score file for writing, or return an error message.
open(SCORES, ">> $ScoreFile") || (printMessage(0) &&
                                  die "Error opening $ScoreFile");
# Save the score in a pipe-delimited text file.
print SCORES join('|', $input->{username}, $input->{score}), "\n";
# Return the result in XML.
printMessage(1);
# Subroutine to output the result in XML.
sub printMessage {
   my $value = shift;
   my $message = {};
```

```
    $message->{success} = $value;
    print XMLout($message, keeproot => 1, rootname => 'success');
}
```

如果使用ColdFusion，看下面的例子代码，将它保存为*gamescores.cfm* ：

```
<cfsilent>
<cfsetting enablecfoutputonly="Yes">
<cfset success = 0>
<cftry>
    <!--- XML packet sent by Flash. --->
    <cfset scores_xml = XmlParse( getHTTPRequestData(    ).content ) >
    <!--- Parse out the XML packet sent from Flash. --->
    <!--- Grab the username and score from the XML document and save as
            local variables so they are easier to work with. // --->
    <cfset username = scores_xml.gamescore.username.XmlText >
    <cfset score = scores_xml.gamescore.score.XmlText >
    <!--- Append the latest score to our scores file. This could also be
            stored in the database or another XML document. // --->
    <cffile action="APPEND" file="#ExpandPath( 'scores.txt' )#"
    output="#username#|#score#|#getHTTPRequestData(    ).content#" addnewline="Yes">
    <cfset success = 1 >
    <cfcatch type="Any">
        <cfset success = 0 >
        <cffile action="APPEND" file="#ExpandPath( 'attempts.txt' )#" output="ERROR"
        addnewline="Yes">
    </cfcatch>
</cftry>
</cfsilent>
<cfcontent type="text/xml">
<cfoutput><?xml version="1.0" ?><success>#success#</success></cfoutput>
<cfsetting showdebugoutput="no" enablecfoutputonly="No">
```

下面是PHP脚本，将它保存为*gamescores.php*:

```php
<?php
// Read In XML from Raw Post Data.
$xml = $GLOBALS['HTTP_RAW_POST_DATA'];
// Process XML using DOM PHP extension.
$document = xmldoc($xml);
// Read root element <gameinfo>.
$rootElement = $document->root(    );
// Read child nodes <username> and <score>.
$childNodes = $rootElement->children(    );
$data = "";
// Loop through child nodes and place in array.
foreach($childNodes as $childNode){
    // Add data to array;
    $name = $childNode->tagName(    );
    $value = $childNode->get_content(    );
    $data[$name] = $value;
}
// Append data to scores.txt ( format: username|score )
$fp = fopen("scores.txt","a+");
$dataString = $data['username'] . "|" . $data['score'] . "\n";
fputs($fp,$dataString,strlen($dataString));
fclose($fp);
// Return success code to Flash
echo "<success>1</success>";
?>
```

20.14. 搜索XML

问题

我想根据某种规则搜索出XML对象的节点或属性

解决办法

使用 E4X 语法和XML对象的过滤器来筛选出特定的值

讨论

这一章讨论了如何用E4X 语法读写XML对象，通过E4X的*XPath* 来搜索XML文档，E4X称得上是最简单的也是最强大的工具，如果你熟悉*XPath*的话，可用E4X的高级特性（如果过滤器）它可根据布尔表达式筛选出指定元素节点。

现在我们开始练习，首先创建一个XML：

```
var foodgroup:XML = <foodgroup>
                        <fruits>
                          <fruit color="red">Apple</fruit>
                          <fruit color="orange">Orange</fruit>
                          <fruit color="green">Pear</fruit>
                          <fruit color="red">Watermelon</fruit>
                          <servings>3</servings>
                        </fruits>
                        <vegetables>
                          <vegetable color="red">Tomato</vegetable>
                          <vegetable color="brown">Potato</vegetable>
                          <vegetable color="green">Broccoli</vegetable>
                          <servings>2</servings>
                        </vegetables>
                      </foodgroup>;
```

如果你事先知道元素节点名称，可直接通过点操作符定位，例如，要返回<fruit>元素节点集合，可使用下面的表达式：

```
var fruitList:XMLList = foodgroup.fruits.fruit;
```

如果只关心某个<fruit> 元素节点，可根据索引值访问：

```
var theApple:XML = foodgroup.fruits.fruit[0];
```

如果你不知道某个节点的完整路径，可使用双点操作符来定位，例如，下面的表达式返回所有

<vegetable>节点：

```
var vegetableList:XMLList = foodgroup..vegetable;
```

(*) 即为“任意节点”，如下面的表达式返回所有<servings>元素节点：

```
var servings:XMLList = foodgroup.*.servings;
```

@ 符号用来访问属性，下面的例子返回<fruit> 节点下的color属性集合：

```
var colorValues:XMLList = foodgroup.fruits.fruit.@color;
```

好，现在我们看一下过滤器，使用（条件）语法来筛选特定的元素节点，条件表达式为布尔值，过滤器作用在*XML* 或 *XMLList* 对象上。

例如，想筛选出<fruit>元素所有color属性为red的节点集合，可设置表达式为@color == "red"

```
/* 显示:
<fruit color="red">Apple</fruit>
<fruit color="red">Watermelon</fruit>
*/
trace( foodgroup..fruit.( @color == "red" ) );
```

在这个例子中的表达式分为两部分：

foodgroups..fruit 部分返回所有<fruit> 元素节点。

过滤器应用在<fruit>节点的*XMLList*集合上并产生新的*XMLList* 对象。

上面的例子选出color属性为red的<fruit>元素节点，但是如果我想选出所有color属性为red的元素节点呢？这时可使用*号选出任何节点其color属性red的元素节点：

```
/* 显示:
<fruit color="red">Apple</fruit>
<fruit color="red">Watermelon</fruit>
<vegetable color="red">Tomato</vegetable>
*/
trace( foodgroup..*.( hasOwnProperty( "@color" ) && @color == "red" )  );
```

上面的例子中，hasOwnProperty 检查元素是否有color属性，如果有在检查其值是否为 red 。.

过滤不仅仅用于筛选属性，还可筛选元素节点，例如下面的例子显示<fruit>元素节点的<name>元素节点的文本值为Apple的color属性值：

```
var fruits:XML = <fruits>
                    <fruit color="red">
                      <name>Apple</name>
                    </fruit>
                    <fruit color="orange">
```

```
                    <name>Orange</name>
                </fruit>
                <fruit color="green">
                    <name>Pear</name>
                </fruit>
                <fruit color="red">
                    <name>Watermelon</name>
                </fruit>
            </fruits>;
// 显示: red
trace( fruits.fruit.(name == "Apple").@color );
```

可以看到过滤器的功能是如此强大，如果和正则表达式联合使用，那将会更强大，如下面的例子使用正则表达式来找出<fruit>节点<name>节点的文本节点值中包含原因字母的集合：

```
var fruits:XML = <fruits>
                <fruit color="red">
                    <name>Apple</name>
                </fruit>
                <fruit color="orange">
                    <name>Orange</name>
                </fruit>
                <fruit color="green">
                    <name>Pear</name>
                </fruit>
                <fruit color="red">
                    <name>Watermelon</name>
                </fruit>
            </fruits>;
/* 显示:
<fruit color="red">
    <name>Apple</name>
</fruit>
<fruit color="orange">
```

```
    <name>Orange</name>
</fruit>
*/
trace( fruits.fruit.( /^[aeiouAEIOU].*/.test( name ) ) );
```

上面的代码创建了一个正则表达式（/之间的内容/），表示起始字母为元音字母，大小写无关，后面跟随任意字母。*test()* 方法测试 name 元素节点的文本节点值是否符合要求。

20.15. 在XML中使用HTML和特殊字符

问题

我想在XML使用HTML或其他一些特殊字符.

解决办法

使用CDATA标签

讨论

在XML中包含的特殊字符需用特殊方式进行处理，例如<和>在XML中作为分隔符，如果你直接在XML文档中的文本内容中使用它们，则会导致语法分析错误，例如：

```
<example>a < b</example>
```

虽然 a < b 是作为文本节点内容，但是<符号在XML文档中是有特殊意义的，因此这将会导致解析错误，另一个普遍问题就是在XML中使用HTML，例如：

```
<htmlExample><a href="http://www.darronschall.com">Darron</a></htmlExample>
```

这里的HTML是作为XML而不是字符串，上面的XML以<htmlExample>为根节点，<a>为子节点且其文本节点值为Darron。

在这些情况必须使用CDATA 标签，把括号中的数据只作为字符串显示。

CDATA标签以<![CDATA[开始，以]]> 结束.，上面的例子可改写成这样：

```
<example><![CDATA[a < b]]></example>
<htmlExample><![CDATA[<a href="http://www.darronschall.com">Darron</a>]]></htmlExample>
```

当 XML 被解析时，CDATA 标签中的内容原封不动。

21.0. 简介

远程过程调用(RPCs)是一种建立分布式应用程序技术，RPC技术是大多数Flash平台应用程序必不可少的技术之一，比如你使用RPC从Flash端发送数据给服务器或者接收来自服务器的数据并在客户端显示，RPC有多种解决方案，不过有两种方式最普遍：web services 和 Flash Remoting。

本书所指的web services，主旨是指简单对象访问协议(SOAP)，服务器之间通过Web services通信所采用的数据传输协议，SOAP会序列化复杂数据类型，当你在客户端调用服务端方法并传递参数（数字，字符串，布尔或复杂类型object）时，服务端方法也可以返回复杂数据给客户端，比如数组，日期甚至是自定义数据类型。现在SOAP web services 几乎在所有平台上都支持，包括Java, ColdFusion, PHP, .NET, 和 Perl ，但是Flash播放器没有内建web services支持，也不理解SOAP。

不过Flash播放器能在HTTP上通讯，能解析XML数据，实际上SOAP web services也是在HTTP上通讯且SOAP协议也是以XML为基础的协议，因此用ActionScript完全可以调用web services方法。

Flash Remoting 就是种类似于web services的技术，他有如下特点：

- Flash Remoting通过HTTP通讯，不过采用的协议不是SOAP，而是一种二进制数据协议，称之为Active Messaging Format (AMF)。因为AMF数据报是二进制的，这样可以传输更多的数据，效率更高，因此 Flash Remoting 速度要比其他web services快。
- Flash Player.支持Flash Remoting。
- Flash Remoting 有Java, ColdFusion, .NET, 和 Perl版本，应用很广泛。

Flash Remoting 和 Web services 都采用异步通讯，都能创建出优秀的 C/S 应用程序。

21.1. 调用Web Services方法

问题

我该如何调用web service 方法

解决办法

使用 *mx.rpc.soap.WebService* 对象，调用*WebService*对象方法

讨论

前面已经提到，Flash播放器没有内建web services支持，但是Flex framework 提供了一个解决方案，这一节将讨论如何使用Flex 2 提供的web services解决方案。

Flex framework 包含*mx.rpc.soap.WebService*，该类可调用web services方法，首先创建*WebService*对象，如下：

```
var webService:WebService = new WebService(    );
```

每个web service都有一个Web服务描述语言(WSDL)，通过*WebService*对象的wsdl属性进行定位：

```
webService.wsdl = "http://www.rightactionscript.com/webservices/FlashSurvey.php?wsdl";
```

在调用方法之前，必须先用*loadWSDL()* 方法读取wsdl数据：

```
webService.loadWSDL(    );
```

loadWSDL() 方法是异步调用的，因此需要监听是否WSDL数据已经读取完毕，当数据读取完成时*WebService* 对象会发出*mx.rpc.soap.LoadEvent* 事件，如：

```
webService.addEventListener(LoadEvent.LOAD, onWSDL);
```

当WSDL数据读取后，就可以调用*WebService*对象方法了，WSDL URL 指向真实的web service ，其有个方法叫*getAverages()* ：

```
webService.getAverages(    );
```

远程方法还可接收参数，这跟调用本地方法没什么两样。例如有个远程方法叫*takeSurvey()*，接收两个整数参数，调用如下：

```
webService.takeSurvey(10, 15);
```

注意 Web services 方法调用也是异步的，也就是说不一定马上得到返回结果。

21.2. 处理Web Services调用结果

问题

我想接收web services 方法的返回值

解决办法

监听web services 对象的 result 事件

讨论

Web services 方法的类型实际上是*mx.rpc.soap.Operation*，当web services 方法返回值时，方法对象会发出 *mx.rpc.events.ResultEvent* 事件，要想处理这个事件可注册监听器，例如，webService有个方法叫*getAverages()*，可这样注册监听器：

```
webService.getAverages.addEventListener(ResultEvent.RESULT, onGetAverages);
```

调用写法和其他方法一样：

```
webService.getAverages(    );
```

当进入处理函数时，会传进来一个*ResultEvent* 参数，*ResultEvent* 类定义了一个叫result 的属性，它包含返回值，因为*getAverages()* 方法返回一个关联数组，包含两个属性：flash 和 actionscript：

```
private function onGetAverages(event:ResultEvent):void {
    textArea.text = "The averages for Flash and ActionScript are " +
event.result.flash + " and " + event.result.actionscript;
}
```

21.3. 处理Web Services异常

问题

我该如何处理web service引发的异常呢.

解决办法

监听 fault 事件

讨论

当web services 引发异常时，方法会发出fault事件，类型为*mx.rpc.events.FaultEvent*，下面的代码注册了fault 事件处理函数：

```
webService.addEventListener(FaultEvent.FAULT, onWebServiceFault);
```

FaultEvent 类定义了一个fault属性，类型为 *mx.rpc.Fault*. *Fault* 对象返回有关异常的信息，包含如faultCode, faultDetail, faultString, 和 rootCause 属性，下面的例子用 *Alert* 显示异常信息：

```
private onWebServiceFault(event:FaultEvent):void {
    var fault:Fault = FaultEvent.fault;
    var message:String = "An error occurred. The details are as follows\ncode: " + fault.faultCode;
    message += "\ndetail: " + faul.faultDetail;
    Alert.show("Web Service Error", message);
}
```

21.4. 调用Flash Remoting方法

问题

我想调用Flash Remoting 方法

解决办法

使用 *NetConnection* 对象连接Flash Remoting网关并用 *call( )* 方法调用方法

讨论

Flex和Flash都有ActionScript APIs 来调用Flash Remoting 方法，不过这一节将讨论用最底层的解决办法。

所有Flash Remoting 方法都基于*flash.net.NetConnection* 类，首先要创建*NetConnection* 对象：

```
var connection:NetConnection = new NetConnection( );
```

下一步调用*connect( )*方法，传递进Flash Remoting网关的URL，例如：

```
connection.connect("http://localhost:8500/flashservices/gateway/");
```

下面的代码连接AMFPHP Flash Remoting 网关：

```
connection.connect("http://www.rightactionscript.com/flashremoting/gateway.php");
```

connect( ) 方法并不会立即连接网关URL，如果URL不合法或服务器发生异常，都不会收到错误直到真正调用方法之后。

下一步就是用*NetConnection*对象的*call( )*方法调用Flash Remoting方法，*call( )*方法需要两个参数，第一个参数指定方法名称和路径，第二个参数指定响应处理函数，如果不需要处理函数，可直接设为null。

第一个参数用点分隔符格式的路径：

```
package.ServiceObject.method
```

如果用ColdFusion 组件(CFC)，可访问http://localhost:8500/com/oreilly/as3cb/Example.cfc, 定义的方法叫*test( )* ：

```
connection.call("com.oreilly.as3cb.Example.test", null);
```

可以继续在后面添加参数，例如*test( )* 方法接收一个数字和字符串作为参数，可这样写：

```
connection.call("com.oreilly.as3cb.Example.test", null, 15, "abcd");
```

21.5. 处理Flash Remoting响应

问题

我想接收Flash Remoting方法返回值

解决办法

使用 *Responder* 对象

讨论

第21.1章 讨论了如何使用*NetConnection*对象的调用Flash Remoting方法，*call()*方法的第二个参数指定处理函数，如果为null则不接收远程方法返回，如果要处理响应，可使用 *flash.net.Responder* 对象。

Responder 构造器可传入两个引用参数，分别为处理返回值和处理异常：

```
var responder:Responder = new Responder(onResult, onError);
```

当返回结果处理函数被调用时，会传进一个为返回值的参数：

```
private function onResult(returnValue:Datatype):void {
}
```

异常处理函数会传进一个包含错误信息的对象。

下面的例子调用Flash Remoting方法*getAverages()*，使用*trace()* 显示，*getAverages()* 方法返回一个关联数组，包含两个属性：flash 和 actionscript：

```
package {
    import flash.net.NetConnection;
    import flash.net.Responder;
    public class Example {
        private var _connection:NetConnection;
        public function Example( ) {
            _connection = new NetConnection( );
            _connection.connect("http://www.rightactionscript.com/flashremoting/
gateway.php");
            var responder:Responder = new Responder(onResult, onError);
            _connection.call("FlashSurvey.getAverages", responder);
        }
        private function onResult(result:Object):void {
            trace(result.flash + " " + result.actionscript);
```

```
        }
        private function onError(error:Object):void {
            trace(error.description);
        }
    }
}
```

22.0. 简介

ExternalInterface 类允许 Flash 播放器以异步的方式与宿主程序进行通信，宿主程序一般指的是 Web 浏览器，这一章将重点讨论 ActionScript 如何与 JavaScript 进行通信。

22.1. 调用JavaScript函数

问题

我想用ActionScript调用JavaScript 函数

解决办法

使用 *ExternalInterface.call()*.

讨论

ExternalInterface.call() 方法采用异步调用JavaScript函数的机制，*call()* 方法至少需要一个参数来指明javascript函数名称：

```
ExternalInterface.call("changeTitle");
```

在HTML页面中定义该函数：

```
<script language="JavaScript">
    function changeTitle(title) {
        if(title == undefined) {
            title = "New Title";
```

```
        }
        window.title = title;
    }
</script>
```

如果JavaScript函数本身需要参数，在*call()*方法参数中继续添加，例如，*changeTitle()* 函数接受一个参数：

```
ExternalInterface.call("changeTitle", "ActionScript 3.0 Cookbook");
```

因为*call()* 是异步调用，不会立即返回结果，我们可以把返回结果保存到变量里：

```
var title:String = ExternalInterface.call("getTitle");
```

JavaScript 函数代码如下；

```
<script language="JavaScript">
    function getTitle(    ) {
        return window.title;
    }
</script>
```

ExternalInterface 支持以下浏览器：

- Internet Explorer 5.0+ (Windows)
- Netscape 8.0+ (Windows and Mac OS X)
- Mozilla 1.7.5+ (Windows and Mac OS X)
- Firefox 1.0+ (Windows and Mac OS X)
- Safari 1.3+ (Mac OS X)

如果*ExternalInterface* 不支持浏览器，而你又想调用JavaScript函数，还可使用*flash.net.navigateToURL()* 函数。

navigateToURL() 函数是异步调用，还有它没有返回值，调用JavaScript函数还须用*flash.net URLRequest*对象进行包装，下面的例子调用JavaScript *alert()* 函数：

```
var request:URLRequest = new URLRequest("javascript:alert('example');");
navigateToURL(request);
```

22.2. 调用ActionScript函数

问题

我想用JavaScript调用ActionScript函数

解决办法

使用 *ExternalInterface.addCallback()* 注册ActionScript 函数，然后在JavaScript端进行调用

讨论

ExternalInterface API 允许注册ActionScript函数，以被JavaScript调用。使用静态方法 *addCallback()* 注册ActionScript函数。*addCallback()* 方法接收两个参数：第一个参数为String类型的函数名，浏览器将借此名称得知要调用的函数，第二参数为浏览器调用定义的函数名时要执行的实际ActionScript函数。下面的例子注册了函数displayMessage ，在JavaScript中的别名为showMessage：

```
ExternalInterface.addCallback("showMessage", displayMessage);
```

在JavaScript端需要得到Flash播放器对象引用，浏览器中的Flash播放器有两种类型：ActiveX 和plug-in版本，ActiveX 版本运行在Internet Explorer上，而plug-in 版本运行在其他浏览器上。

ActiveX 版本播放器由HTML页中的<object>标签控制，通过window. *objectId* 让JavaScript得到Flash播放器引用，*objectId* 是<object>标签的id属性值，比如 <object> 标签的id属性为example，那么ActiveX 播放器的引用就是 window.example 。

plug-in 版本播放器由HTML页中的<embed>标签控制，通过window.document.*embedName* 让JavaScript得到Flash播放器引用，*embedName* 是<embed>标签的name属性值，比如<embed>标签的name属性为example，那么plug-in播放器引用就是 window.document.example 。

一般情况下，我们并不知道用户使用什么版本的Flash播放器，有个方法是通过JavaScript的navigator.appName 来检测用户浏览器的类型：

- 如果navigator.appName 包含Microsoft关键字，那么用户使用的就是Internet Explorer，也就是ActiveX 播放器。
- 如果navigator.appName 不包含Microsoft关键字，也就意味着是plug-in 版本播放器。

下面的JavaScript代码用户检测播放器类型：

```
<script language="JavaScript">
var flashPlayer;
function detectFlashPlayer(    ) {
    if(navigator.appName.indexOf("Microsoft") != -1) {
        flashPlayer = window.objectId;
    }
    else {
```

```
            flashPlayer = window.document.embedName;
        }
}
</script>
```

下一步，在<body>的onLoad属性中调用*detectFlashPlayer()* 函数：

```
<body onLoad="detectFlashPlayer">
```

变量包含当前Flash播放器的对象引用。

可通过flashPlayer变量来调用注册的ActionScript函数，例如，如果注册函数的别名为showMessage，用下面的语句进行调用：

```
flashPlayer.showMessage(    );
```

还可以传递参数，如果*ExternalInterface* 定义的showMessage 接收一个字符型参数，可这样写：

```
flashPlayer.showMessage("example message");
```


22.3. 从HTML中传递参数给Flash

问题

我想把HTML中的变量作为参数传递给SWF.

解决办法

使用 FlashVars.

讨论

FlashVars 提供一个解决方案用户把HTML变量作为参数传递给SWF。这点在传递一些简单的数据给SWF时是非常有用的功能，例如当web services URL发生变化时你可能需要传递一个新的URL给SWF，这样就可避免再次重新编译SWF。

FlashVars 方案有两部分组成，一部分通过HTML实现，另一部分由ActionScript实现。

HTML部分需要在<object>标签中添加一个<param name="FlashVars">标签，例如下面的FlashVars 定义了两个键值对：url1 和 url2：

```
url1=http://www.example.com&url2=http://www.sample.com
```

在ActionScript方面，可通过任何可视化组件都有的root.loaderInfo.parameters 属性得到FlashVars传递进来的参数，root.loaderInfo.parameters 属性是一个关联数组，例如根据上面例子的变量值，root.loaderInfo.parameters 属性将得到两个键值：url1 和 url2 。

通过JavaScript的FlashVars，我们可传递一个字符串序列给SWF，下面的例子演示如何编写

<object> 和<embed> 标签以及通过FlashVars传递字符串序列给SWF：

```
// Retrieve the query string, and assign it to a variable.
var parameters = window.location.search.substr(1);
var objectEmbed = '<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" id="Example" width="100%" height="100%" codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab">';
objectEmbed += '<param name="movie" value="Example.swf" />';
objectEmbed += '<param name="quality" value="high" />';
objectEmbed += '<param name="bgcolor" value="#869ca7" />';
objectEmbed += '<param name="allowScriptAccess" value="sameDomain" />';
objectEmbed += '<param name="FlashVars" value="' + parameters + '" />';
objectEmbed += '<embed src="Example.swf" quality="high" bgcolor="#869ca7" width="100%" height="100%" name="Example" align="middle" play="true" loop="false" quality="high" allowScriptAccess="sameDomain" type="application/x-shockwave-flash" pluginspage="http://www.macromedia.com/go/getflashplayer" FlashVars="' + parameters + '"></embed>';
objectEmbed += '</object>';
document.write(objectEmbed);
```

FlashVars 是 Flash 播放器一个重要的功能，但是仅仅用 FlashVars 是不够的，它只能传递一些简单的数据，如果要传递复杂的数据，可使用 *URLLoader* 对象。

23.0. 简介

在Flash播放器8以前的版本中并没有提供管理文件的功能，也没有一种机制用于上传和下载文件，因此大多数Web程序都是基于HTML的方式上传和下载文件，基于Flash的应用程序必须自己来实现上传和下载文件功能，从Flash播放8开始，系统提供了一套新的APIs 来支持文件的上传和下载。

Flash 播放器允许用户浏览本地文件，并使用 *FileReference* 和 *FileReferenceList* 类上传和下载文件，本章将讨论这些 APIs 。

23.1. 下载文件

问题

我想让用户从服务器上下载文件

解决办法

使用*FileReference* 对象的 *download()* 方法

讨论

flash.net.FileReference 类定义了一个 *download()* 方法允许用户通过URL下载文件。当Flash播放器调用*download()*方法时，它试图打开一个对话框，标题为"Select location for download."，这个对话框使用标准的系统对话框让用户选择文件保存路径。

在调用*download()*方法之前，先构造一个*FileReference* 对象，如：

```
var fileReference:FileReference = new FileReference(    );
```

download() 方法至少需要一个*URLRequest* 对象作为参数，指定下载文件的路径，下面的例子打开保存对话框保存下载的*example.txt* 文件：

```
var urlRequest:URLRequest = new URLRequest("example.txt");

fileReference.download(urlRequest);
```

文件路径还可以是绝对路径：

```
var urlRequest:URLRequest = new URLRequest("http://www.myexamplesite.com/example.txt");

fileReference.download(urlRequest);
```

保存对话框允许对下载的文件进行重命名，默认是与服务器文件的名称相同，如上面的例子，保存对话框显示的名称为*example.txt*。但在大多数情况下，我们喜欢重命名文件，比如一个有服务端脚本产生的静态文件采用统一的命名规则进行命名(如 *R7AS82892KHWI014.jpg*) 可能它不

会显示出这个名字，因为这是由服务端脚本动态生成，如果*URLRequest*对象指向这个脚本那么保存对话框显示的可能就是脚本的名字(如., *script.cgi*)，这样很不友好，也将导致文件无法被系统识别，因为文件扩展名并不正确。

download() 方法的第二个参数就是指定显示在保存对话框的文件名，下面的例子演示用户下载一个动态生成的图片文件，默认的文件名为(*script.cgi*)，这里指定保存的文件名为*example.jpg*:

```
var urlRequest:URLRequest = new URLRequest("script.cgi");
fileReference.download(urlRequest, "example.jpg");
```

download() 方法最好放在try...catch 语句中执行，因为该方法可能会抛出异常，主要两种异常: *IllegalOperationError*和*SecurityError*。当保存对话框已经打开的情况下调用*download()*方法会抛出*IllegalOperationError*，*SecurityError*异常是由于SWF不允许下载所导致。

```
try {
    fileReference.download(urlRequest, fileName);
}
catch (illegalOperation:IllegalOperationError) {
    // code to handle an illegal operation error
}
catch (security:SecurityError) {
    // code to handle a security error
}
```

还有些不太常见的异常，如*ArgumentError*和*MemoryError*。*ArgumentError*是由于*URLRequest* 的data属性不是*URLVariables*。 虽然*URLRequest*的data属性可设置为二进制数组，但是*download()*方法只有在data属性为*URLVariables*时才有效。memory 异常比较少见，有两种可能会导致此异常：当*URLRequest*设置为GET请求而且System.useCodePage为true，Flash播放器将不能把Unicode转换到字节字符格式，还有可能就是Flash播放器没有足够的内存打开保存对话框。

像*SecurityErrorEvent* 和*IOErrorEvent*.异常必须监听异常处理函数，如下面的例子为*IOErrorEvent*添加处理函数：

```
fileReference.addEventListener(IOErrorEvent.IO_ERROR, onIOError);
```

23.2. 检测用户是否已选择了下载文件

问题

我想知道用户是否已选择了文件即将进行下载

解决办法

监听select事件和cancel事件

讨论

download() 方法本身没有暂停执行功能，一旦*download()*方法被调用，Flash播放器就会试图打开保存对话框，要么成功打开对话框，要么抛出异常，而Flash播放器则继续执行下一行代码，也就是或系统并不知道用户是否已经选择了文件并点击保存按钮，因此需要监听select事件确定用户何时点击了保存按钮，select事件的类型为*Event*，使用Event.SELECT常量作为参数：

```
fileReference.addEventListener(Event.SELECT, onSelectFile);
```

在select事件中，通过*FileReference*对象的name属性可获得用户选择的文件名：

```
private function onSelectFile(event:Event):void {
    trace(event.target.name);
}
```

当用户也可能点击保存对话框的Cancel按钮，如果用户点击了Cancel按钮，文件将不会被下载，对话框将关闭：

```
fileReference.addEventListener(Event.CANCEL, onCancelDialog);
```


23.3. 监视文件下载进度

问题

我想知道文件的下载进度

解决办法

监听progress事件

讨论

我们可以通过progress事件监视文件的下载进度，在下载过程中*FileReference*对象会不断的发出*ProgressEvent*类型事件：

```
fileReference.addEventListener(ProgressEvent.PROGRESS, onFileProgress);
```

progress事件对象有两个属性,bytesLoaded和bytesTotal属性，返回当前已下载字节数和总字节数,

看下面的例子：

```
private function onFileProgress(event:ProgressEvent):void {
    fileProgressField.text = event.bytesLoaded + " of " + event.bytesTotal + " bytes";
}
```

当下载完成时*FileReference*对象发出complete事件，类型为*Event*：

```
fileReference.addEventListener(Event.COMPLETE, onFileComplete);
```

23.4. 浏览本地文件

问题

我想浏览本地文件以便上传

解决办法

使用*FileReference*或*FileReferenceList*对象的*browse()*方法

讨论

使用*FileReference*或*FileReferenceList*对象的*browse()*方法可以打开一个对话框用于浏览本地磁盘文件，唯一不同的是*FileReference*对象的*browse()*方法只能选一个文件，而*FileReferenceList*对象可以选多个文件：

```
fileReference.browse( );
```

和*download()*方法一样，*browse()*方法也可能抛出异常，因此在调用时最好放在try/catch语句中执行并处理可能的异常，导致illegal operation error异常可能有两种可能：

一次只能打开一个浏览对话框，如果再次调用*browse()*，就会抛出*IllegalOperationError*。

如果用户对全局Flash播放器设置为不允许文件浏览，则调用*browse()*方法时抛出该异常。

下面的代码段演示如何在TRy/catch块中处理*IllegalOperationError*异常：

```
try {
    fileReference.browse(    );
}
catch (illegalOperation:IllegalOperationError) {
    // code to handle error
}
```

23.5. 过滤浏览对话框显示的文件

问题

我想让浏览对话框只显示特定类型的文件

解决办法

传递一个*FileFilter*对象数组作为*browse()*方法参数

讨论

默认下*browse()*方法打开的对话框显示用户系统中的所有文件，可以通过设置过滤器只显示特定类型的文件，比如只显示图形文件或文本文件，设置的方法是把*flash.net.FileFilter*对象数组作为参数传递给*browse()*方法。

*FileFilter*构造器至少需要两个参数：

第一个参数决定在下拉列表中显示什么文件类型：
第二个参数决定显示的文件扩展名。

多个文件扩展名需用分号分开，如下面的例子只显示三种类型的图形文件：

```
var fileFilter:FileFilter = new FileFilter("Images", "*.png;*.gif;*.jpg");
```

可以在调用*browse()*方法时进行过滤设置：

```
fileReference.browse([fileFilter]);
```

下面的代码只设置了一个过滤器，还可以一次传入多个过滤器，如：

```
var fileFilter1:FileFilter = new FileFilter("Images", "*.png;*.gif;*.jpg");
var fileFilter2:FileFilter = new FileFilter("Documents", "*.txt;*.doc;*.pdf;*.rtf");
var fileFilter3:FileFilter = new FileFilter("Archives", "*.zip;*.tar;*.hqx");
var fileFilter4:FileFilter = new FileFilter("All", "*.*");
_fileReference.browse([fileFilter1, fileFilter2, fileFilter3, fileFilter4]);
```

在设置过滤器后，调用*browse()*方法可能会抛出*ArgumentError*异常，这是由于*FileFilter*对象的格式错误造成的：
文件过滤器同样适用于 *FileReferenceList* 对象的 *browse()*方法

23.6. 监测用户是否选择了文件准备上传

问题

我想知道用户是否通过浏览对话框选择了文件

解决办法

监听select事件和cancel事件

讨论

当用户选择了文件并点击了Open按钮后*FileReference*对象会发出select事件，类型为*Event*，可通过Event.SELECT常量注册监听器：

```
fileReference.addEventListener(Event.SELECT, onSelectFile);
```

当用户选择了文件后，关于文件的信息（如文件名，大小，创建时间等）都会保存在*FileReference*对象里：

```
selectedFileTextField.text = fileReference.name;
```

如果用户点击了Cancel按钮，*FileReference*对象会发出cancel事件，类型为*Event*。可通过Event.CANCEL常量注册监听器：

```
fileReference.addEventListener(Event.CANCEL, onCancelBrowse);
```

这两个事件同样适用于 *FileReferenceList* 对象

23.7. 上传文件

问题

我想让用户上传文件

解决办法

使用*FileReference*对象的*upload()*方法

讨论

*FileReference*对象的*upload()*方法允许使用服务端脚本通过HTTP(s)上传文件，*upload()*方法至少需要一个为*URLRequest*类型的对象作为参数，用于指定服务端脚本的URL：

```
var urlRequest:URLRequest = new URLRequest("uploadScript.cgi");
```

```
fileReference.upload(urlRequest);
```

所有上传使用POST传输方式，Content-Type为multipart/form-data。默认下Content-Disposition设置为Filedata，因为脚本需要知道Content-Disposition值以便读取文件数据。如果脚本需要Content-

Disposition其他值，可通过*upload()*方法第二个参数设置：

```
fileReference.upload(urlRequest, "UploadFile");
```

如果是调用*FileReferenceList*对象，那根据fileList属性必须为每个文件调用一次*upload()*方法，*FileReferenceList*对象的fileList属性就是*FileReference*对象数组。
*upload()*方法抛出的异常和 *download()*方法相同，比如都会抛出 security 异常事件和 IO 异常事件。

23.8. 监视文件上传进度

问题

我想知道文件的上传进度

解决办法

监听progress事件

讨论

和监视下载进度一样，当文件上传时 *FileReference* 对象发出 progress 事件，完成时发出 completed 事件，请看 第 23.3 节 。

24.0. 简介

Socket 套接字连接允许Flash播放器通过指定的端口与服务器通信，socket连接与其他通信技术最大的不同是socket连接在数据传输完成后不会自动关闭。

当socket连接创建后，连接会一直保持，直到客户端（Flash播放器）和服务端主动关闭，因此服务器可在任何时间不用客户端请求即可发送数据给客户端。

Socket连接被普遍用于创建多用户应用程序，比如说一个在线聊天室，它有一个服务端程序和无数个Flash客户端组成。每次客户端发送消息给服务器，服务器检测那些用户可以收到这些消息并把消息传给指定客户端，这种情况下接收客户端并没有提前请求数据而是通过服务器主动推送数据的。当客户端关闭时，服务端提示其他客户端某客户端已离线。

Flash播放器提供了两种类型的socket连接。一种是早期版本就有的*XMLSocket*，Flash播放器9增加了二进制socket连接。

使用*flash.net.XMLSocket*类创建XML数据格式的socket连接，使用*flash.net.Socket*类创建二进制数据格式socket连接。

XML socket连接以XML数据报交换数据，二进制socket连接是ActionScript 3.0新增的功能，相比之下更低级，但功能很强大，几乎可以连接任意类型的socket服务端程序。例如二进制sockets 可连接邮件服务端程序(POP3, SMTP, 和IMAP), 新闻服务器(NNTP), 聊天室服务器或远程桌面VNC服务器(RFB)。

不管是哪种类型的 socket 连接，其通信方式都是异步的，也就是说你不能直接从 socket 连接中读取数据，而是通过事件处理函数进行读取处理。

24.1. 连接Socket服务器

问题

我想连接socket服务器

解决办法

使用*Socket.connect( )*或*XMLSocket.connect( )*方法建立连接并监听connect事件确定连接是否建立。

讨论

要连接socket服务器，首先要知道域名或IP地址，还要知道端口，不管是使用*Socket*还是*XMLSocket*，连接步骤是一样的，都是用*connect( )*方法进行连接，该方法接受两个参数：

host

指定域名或IP地址，如www.example.com或192.168.1.101。

port

数字，指定连接的端口号，必须大于1024，如果小于1024则需服务器提供策略文件允许。

因为是异步通信，*connect( )*方法不会等待结果而是继续执行下面的语句，因此需要注册事件监听器来获取连接结果。

注册事件监听器必须在调用*connect( )*方法之前，当连接成功时connect事件就会触发，下面的例子演示连接本机2900端口：

```
package {
   import flash.display.Sprite;
   import flash.events.*;
   import flash.net.Socket;
   public class SocketExample extends Sprite {
      private var socket:Socket;
      public function SocketExample(    ) {
         socket = new Socket(    );
         // Add an event listener to be notified when the connection
         // is made
         socket.addEventListener( Event.CONNECT, onConnect );
         // Connect to the server
         socket.connect( "localhost", 2900 );
```

```
        }
        private function onConnect( event:Event ):void {
            trace( "The socket is now connected..." );
        }
    }
}
```

如果使用*XMLSocket*，代码也基本上相同，代码如下：

```
package {
    import flash.display.Sprite;
    import flash.events.*;
    import flash.net.XMLSocket;
    public class SocketExample extends Sprite {
        private var socket:XMLSocket;
        public function SocketExample(    ) {
            socket = new XMLSocket(    );
            // Add an event listener to be notified when the connection is made
            socket.addEventListener( Event.CONNECT, onConnect );
            // Connect to the server
            socket.connect( "localhost", 2900 );
        }
        private function onConnect( event:Event ):void {
            trace( "The xml socket is now connected..." );
        }
    }
}
```

如果连接失败，可能的异常有：runtime error，ioError，securityError ，

记住，当用socket连接主机时，要遵循Flash Player安全沙漏规则：

swf 和主机必须在同一个域；
网络上的*swf*不能连接本地服务器；
本地的*swf* 不能访问任何网络资源；
要允许域名交叉访问或连接低于1024的端口，需要提供cross-domain 策略文件。

如果*Socket*或*XMLSocket*对象要使用cross-domain策略文件，可通过方法*flash.system.Security.loadPolicyFile()*读取：

```
Security.loadPolicyFile("http://www.rightactionscript.com/crossdomain.xml");
```

cross-domain策略文件例子：

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
    <allow-access-from domain="*" to-ports="80,110" />
</cross-domain-policy>
```

24.2. 发送数据

问题

我要发送数据给socket服务器

解决办法

对于*Socket*对象，是使用write方法(*writeByte()*, *writeUTFBytes()*, 等等) 把数据写入到缓冲区，再通过*flush()*方法发送数据，对于*XMLSocket*对象使用*send()*方法。

讨论

*Socket*和*XMLSocket*类各自定义了不同的APIs来发送数据，首先看一下*Socket*的API ：

当使用*Socket*对象发送数据时必须先把数据写到缓冲区，*Socket*类定义了一系列方法用于写数据，每个方法都是写入不同类型的数据，这些方法是*writeBoolean()*, *writeByte()*, *writeBytes()*, *writeDouble()*, *writeFloat()*, *writeInt()*, *writeMultiByte()*, *writeObject()*, *writeShort()*, *writeUnsignedInt()*, *writeUTF()*, 和*writeUTFBytes()*。每个方法都接受一种类型的参数，如*writeBoolean()*接受布尔型参数，*writeByte()*, *writeDouble()*, *writeFloat()*, *writeInt()*, *writeShort()*, 和*writeUnsignedInt()*接受数字参数，*writeObject()*方法接受对象参数，*writeBytes()*方法允许传递一个*ByteArray*类型参数以及偏移量和长度，如下面的写法：

```
socket.writeBytes(byteArray, 0, byteArray.length);
```

*writeUTF()*和*writeUTFBytes()*方法写入字符串，每个方法接受一个字符串参数，*writeUTFBytes()*方法写入字节形式的字符串，*writeUTF()*方法写入字节数字。

*writeMultiByte()*方法也是写入字符串数据，但可以不使用默认字符集，该方法接受两个参数：字符串和指定字符集编码，下面的例子写入Unicode编码的字符串：

```
socket.writeMultiByte("example", "unicode");
```

利用*Socket*对象，完全可以用ActionScript写出一个Telnet和POP客户端，这两个协议都是以ASCII字符为基础的，例如，连接一个POP服务器后，可用下面的代码执行USER命令：

```
// POP servers expect a newline (\n) to execute the preceding command.
socket.writeUTFBytes("USER exampleUsername\n");
```

写入的数据实际上还没发送到服务器上，每个方法都死把数据累积到*Socket*对象上，例如下面的四个代码并没有把数据发送出去：

```
socket.writeByte(1);
socket.writeByte(5);
socket.writeByte(4);
socket.writeByte(8);
```

当要发送数据时，必须调用*flush()*方法，*flush()*方法发送所有的数据并清空缓冲区：

```
socket.flush(  );
```

*XMLSocket*类发送数据就比较简单了，发送数据的方法为*send()*，*send()*方法接受任意类型的数据类型，它会把参数转换为字符串并发送给服务器，一般这个参数是一个XML对象：

```
xmlSocket.send(xml);
```

实际上发送的数据类型是由服务器所决定的，如果服务器接受 XML 数据，那必须发送 XML 数据，如果服务器接受 URL-编码数据，则必须发送 URL-编码数据。

24.3. 接收数据

问题

我想接收socket服务器发送来的数据

解决办法

对于*Socket*实例可通过 ProgressEvent.SOCKET_DATA事件处理函数中读取数据，可用*readByte()*或*readInt()*方法

对于*XMLSocket*实例可通data事件处理函数中读取XML数据

讨论

从socket中接收数据的方法取决于你使用socket类型，*Socket*和*XMLSocket*都可以接收数据，但是两者实现方法有些不同，让我们先看看*Socket*类是如何做的。

正如前面所讲到的,Flash提供的socket通信方式是异步通信，也就是说仅仅创建socket连接并试图读取数据这是不可能的，*read*方法读取数据时并不会等待数据传输过来而立即返回,如果数据还没准备好而去读取数据会导致异常。

当数据准备好了时，socketData事件就会触发，通过注册该事件处理函数，当数据发送过来时就会触发，因此可通过该处理函数读取数据。

为了读取服务器发送来的数据，*Socket*类提供了一系列*read*方法来读取不同类型的数据，例如通过*readByte()*方法读取一个字节，*readUnsignedInt()*方法读取一个无符号整数，具体看下面的表格：

Table 24-1. Socket read methods for various datatypes

方法：返回类型	描述	字节数
readBoolean():Boolean	读取布尔型数据	1
readByte():int	读取一个字节数据	1
readDouble():Number	读取 IEEE 754 双精度浮点数	8
readFloat():Number	读取 IEEE 754 单精度浮点数	4
readInt():in	读取 32 位整数	4
readObject():*	读取 AMF 格式对象	N
readShort():int	读取 16 位整数	2
readUnsignedByte():uint	读取无符号字节	1
readUnsignedInt():uint	读取无符号 32 位整数	4
readUnsignedShort():uint	读取无符号 16 位整数	2
readUTF():String	读取 UTF-8 字符串	n

还有两个方法没有在上面的表格里，它们是*readBytes()*和*readUTFBytes()*，*readBytes()*方法没有返回值，它接受三个参数：

bytes

一个*flash.util.ByteArray* 实例填充读取的数据

offset

一个*uint*值指定读取数据的偏移量，默认为0

length

一个*uint*值表示读取的字节数，默认为0，表示所有的数据

*readUTFBytes()*方法只接受一个参数，表示读取的 UTF-8 字节数，返回一个字符串。

下面的例子代码连接一个socket服务器并读取和显示服务器发送的数据：

```
package {
   import flash.display.Sprite;
   import flash.events.ProgressEvent;
   import flash.net.Socket;
   public class SocketExample extends Sprite {
      private var socket:Socket;
      public function SocketExample(    ) {
```

```
            socket = new Socket(    );
            // Listen for when data is received from the socket server
            socket.addEventListener( ProgressEvent.SOCKET_DATA, onSocketData );
            // Connect to the server
            socket.connect( "localhost", 2900 );
        }
        private function onSocketData( event:ProgressEvent ):void {
            trace( "Socket received " + socket.bytesAvailable + " byte(s) of data:" );
            // Loop over all of the received data, and only read a byte if there
            // is one available
            while ( socket.bytesAvailable ) {
                // Read a byte from the socket and display it
                var data:int = socket.readByte(    );
                trace( data );
            }
        }
    }
}
```

上面的例子中，如果socket服务器发送"Hello"字符串，则输出：

```
//Socket接收5字节的数据：
72
101
108
108
111
```

Socket对象接收的数据都是ASCII编码的文本，我们可以用readUTFBytes( )方法重新构造字符串，readUTFBytes( )方法需要知道有多少个字节需要转换，用bytesAvailable属性指定字节数：

```
var string:String = socket.readUTFBytes(socket.bytesAvailable);
```

*XMLSocket*类和*Socket*类基本类似，两者都要注册监听器检测数据是否接收完毕，但是两者读取数据的方式是不同的。

当数据准备好时 *XMLSocket* 实例发出 data 事件，事件类型为 *flash.events.DataEvent.DATA* ，它

其中的 data 属性包含接收过来的数据。

从服务器返回的数据都是原始的数据，如果你希望以XML进行处理则需先把数据转换为*XML*实例。

下面的代码例子用*XMLSocket*连接本地服务器，端口为2900，连接成功后，发送<*test*>消息给服务器，*onData*事件处理函数处理服务器返回的数据,返回数据为 <response><test success='true'/></response>，注意到该事件的data属性的内容只是字符串数据，需用XML构造器转换位XML实例，最后通过*E4X* 语法输出XML：

```
package {
   import flash.display.Sprite;
   import flash.events.Event;
   import flash.events.DataEvent;
   import flash.net.XMLSocket;
   public class SocketExample extends Sprite {
      private var xmlSocket:XMLSocket;
      public function SocketExample(    ) {
         xmlSocket = new XMLSocket(    );
         // Connect listener to send a message to the server
         // after we make a successful connection
         xmlSocket.addEventListener( Event.CONNECT, onConnect );
         // Listen for when data is received from the socket server
         xmlSocket.addEventListener( DataEvent.DATA, onData );
         // Connect to the server
         xmlSocket.connect( "localhost", 2900 );
      }
      private function onConnect( event:Event ):void {
         xmlSocket.send( "<test/>" );
      }
      private function onData( event:DataEvent ):void {
         // The raw string returned from the server.
         // It might look something like this:
         // <response><test success='true'/></response>
         trace( event.data );
```

```
            // Convert the string into XML
            var response:XML = new XML( event.data );
            // Using E4X, access the success attribute of the "test"
            // element node in the response.
            // Output: true
            trace( response.test.@success );
        }
    }
}
```

24.4. 与socket服务器的状态信号交换

问题

我想与服务器进行信号交换以便知道读取的数据内容是什么以及如何进行处理。

解决办法

创建不同的常量来表示协议状态，用这些常量映射与之对应的状态处理函数，在socketData事件处理函数中通过状态映射表调用对应的状态处理函数。

讨论

连接socket服务器需要经过一个完整的信号交换，通常服务器初始化后发送数据给客户端，客户端进行回应，服务器再回应，这个完整的处理过程一直重复直到信号交换完成，这个连接才算建好了。

就像典型的HTTP连接一样，HTTP协议定义了一系列状态代表不同的传输数据，我们现在建立的socket连接也是最原始的无状态的，同样需要建立各种状态以及维护各种状态的函数功能。

解决办法就是创建不同的状态常量代表服务器发送的不同类型的内容，每个状态关联不同的状态处理函数。

连接一个socket服务器需要的信号交换可能有：

当客户端连接到服务器时，服务器立即回应，发送一个整数代表服务器所支持的协议版本。
客户端返回一个整数表示可以通信的协议版本。
服务器发送8自己的认证询问。
客户端发送认证给服务器。
如果客户端的回应不合法或协议不一致或不是规定的信息则关闭连接。

要实现这个信号交换过程，首先要创建代表不同状态的常量，例如建立如下常量：

```
public const DETERMINE_VERSION:int = 0;
public const RECEIVE_CHALLENGE:int = 1;
public const NORMAL:int = 2;
```

常量设置为什么值并不重要，重要的是这些常量值都应该不同，下一步就是创建代表不同状态的处理函数，如创建*readVersion()*, *readChallenge()*, 和*readNormalProtocol()*，接着要把这些函数与状态常量相关联，如下面的代码：

```
stateMap = new Object(    );
stateMap[ DETERMINE_VERSION ] = readVersion;
stateMap[ RECEIVE_CHALLENGE ] = readChallenge;
stateMap[ NORMAL                    ] = readNormalProtocol;
```

最后就是在socketData事件处理函数中根据当前的状态调用相应的状态处理函数了，创建currentState变量代表当前状态，根据其值调用对应的处理函数：

```
var processFunc:Function = stateMap[ currentState ];
processFunc(    ); // Invoke the appropriate processing function
```

完整代码如下：

```
package {
   import flash.display.Sprite;
   import flash.events.ProgressEvent;
   import flash.net.Socket;
   import flash.utils.ByteArray;
   public class SocketExample extends Sprite {
      // The state constants to describe the protocol
      public const DETERMINE_VERSION:int = 0;
      public const RECEIVE_CHALLENGE:int = 1;
      public const NORMAL:int = 2;
      // Maps a state to a processing function
      private var stateMap:Object;
      // Keeps track of the current protocol state
      private var currentState:int;
      private var socket:Socket;
      public function SocketExample(    ) {
```

```
        // Initialzes the states map
        stateMap = new Object(    );
        stateMap[ DETERMINE_VERSION ] = readVersion;
        stateMap[ RECEIVE_CHALLENGE ] = readChallenge;
        stateMap[ NORMAL                    ] = readNormalProtocol;
        // Initialze the current state
        currentState = DETERMINE_VERSION;
        // Create and connect the socket
        socket = new Socket(    );
        socket.addEventListener( ProgressEvent.SOCKET_DATA, onSocketData );
        socket.connect( "localhost", 2900 );
    }
    private function onSocketData( event:ProgressEvent ):void {
        // Look up the processing function based on the current state
        var processFunc:Function = stateMap[ currentState ];
        processFunc(    );
    }
    private function readVersion(    ):void {
        // Step 1 - read the version from the server
        var version:int = socket.readInt(    );
        // Once the version is read, the next state is receiving
        // the challenge from the server
        currentState = RECEIVE_CHALLENGE;
        // Step 2 - write the version back to the server
        socket.writeInt( version );
        socket.flush(    );
    }
    private function readChallenge(    ):void {
        // Step 3 - read the 8 byte challenge into a byte array
        var bytes:ByteArray = new ByteArray(    );
        socket.readBytes( bytes, 0, 8 );
```

```
            // After the challenge is received, the next state is
            // the normal protocol operation
            currentState = NORMAL;
            // Step 4 - write the bytes back to the server
            socket.writeBytes( bytes );
            socket.flush(    );
        }
        private function readNormalProtocol(    ):void {
            // Step 5 - process the normal socket messages here now that
            // that handshaking process is complete
        }
    }
}
```


24.5. 断开与Socket服务器的连接

问题

我想断开与服务器的连接或通知服务器断开

解决办法

调用*Socket.close()* 或 *XMLSocket.close()* 方法关闭连接，或者监听close事件

讨论

当我们连接一个socket连接后，用完时应该关闭连接，释放资源，如果不关闭这会导致占用端口使其它连接无法建立。

Socket 和*XMLSocket*关闭socket连接的方法是一样的，都为*close()* 方法：

```
// Assume socket is a connected Socket instance
socket.close(    );    // Disconnect from the server
```

调用*XMLSocket*实例的方法：

```
// Assume xmlSocket is a connected XMLSocket instance
xmlSocket.close(    );    // Disconnect from the server
```

close() 方法通知服务器客户端已经断开连接，如果服务器先关闭呢，那客户端该怎么知道呢，可通过监听客户端的close事件，注册事件类型为Event.CLOSE：

```
var socket:Socket = new Socket(    );
socket.addEventListener( Event.CLOSE, onClose );
```

24.6. 处理Socket异常

问题

使用socket如何处理可能引发的异常

解决办法

使用 try/catch 处理I/O 和 (EOF) 异常

讨论

Socket 和*XMLSocket* 类处理异常的方式基本类似，当调用*connect()*方法时，如遇到下面的情况 *Socket* 和*XMLSocket* 对象都抛出*SecurityError*：

.swf 被认为本地非安全
端口号高于655535.

当调用 *send()* (*XMLSocket*) 或 *flush()* (*Socket*)，如果没有事先连接好，则抛出*IOError*，可先通过socket对象的connected属性是否为true，再调用*send()* 或 *flush()*。如下面的代码：

```
if ( socket.connected ) {
    try {
        socket.flush(    );
    }
    catch( error:IOError ) {
        logInstance.write( "socket.flush error\n" + error );
    }
}
else {
    connectToSocketServer(    ); //进行socket连接
}
```

Socket 所有的read 方法都可能抛出*EOFError*和*IOError*，但没有数据可读而读取时引发EOF异常，当socket已关闭而去读取时引发I/O 异常。