



STACK & HEAP

# 关于ActionScript中 那些你不知道的事情

Christophe Herreman  
@herrodus



二翻队 Shinji

探索ActionScript3语言，我们将会发现一些前所未见的，指不定哪天就会用到新东西。又或者你可能会突然惊叫一声“我X！”——搞得在座看官们还以为我XJJ露出来了——但其实你只是看到了一段你觉得挺非主流的ActionScript代码随后疑惑焦虑惊叹羡慕嫉妒恨别人居然是这么用API的，换句话说你觉得原先用着挺别扭的AS语句现在突然变得好用甚至有趣了，其实很显然ActionScript本身还是又酷又强大的！

# 我是谁？

我运营 **Stack & Heap** 公司，位于比利时  
公司专注于Web/RIA相关领域的开发、咨询及培训。

Flex/AIR 认证专家，1999年起爱上Flash

**Spring ActionScript** 和 **AS3Commons** 的创始人

**Apache Flex** 提交者

# 运算符

# 等于 "=="

比较2个值是否相等，当被比较的值类型不同时自动进行转换

```
"hello" == "hello"    // true
"5" == 5              // true
true == 1             // true
false == 0            // true
"true" == true        // false
null == undefined     // true
```

# 严格等于 "==="

比较2个数据的值和数据类型是否都相等

```
"hello" === "hello"    // true
"5" === 5              // compile error
true === 1             // false
false === 0            // false
"true" === true        // false
null === undefined     // false
```

# 等于 "=="

复杂数据类型将基于引用而非值进行比较

```
var a:Array = [1, 2, 3];  
var b:Array = [1, 2, 3];
```

```
a == b    // false
```

```
var c:Array = a;
```

```
a == c    // true
```

# 条件 "?"

又叫“三元”运算符

```
var result:Boolean = (a > b) ? x : y;
```

// 上面这句可以用来替代以下代码，神奇不？

```
var result:Boolean;  
if (a > b) {  
    result = x;  
} else {  
    result = y;  
}
```



# 逻辑或 "||="

```
function (a:Object) {  
    a ||= new Object();  
}
```

// 上面的代码和下面的代码作用是一样的，神奇不？

```
function (a:Object) {  
    if (a === null) {  
        a = new Object();  
    }  
}
```

# 逻辑与 "&&="

```
function toHTMLTag (s:String) {  
    s &&= "<" + s + ">";  
    return s;  
}
```

// 上面的代码和下面的代码作用是一样的，神奇不？

```
function toHTMLTag (s:String) {  
    if (s !== null && (s.length > 0))  
        s = "<" + s + ">";  
    return s;  
}
```

# "as"

转换一个值的数据类型，如果转换失败则返回null

```
"hello" as String    // "hello"  
5 as String          // null  
true as MyClass      // null  
  
String("hello")      // "hello"  
String(5)             // "5"  
MyClass(true)         // Runtime Error
```

# "is" vs "instanceof"

检查一个值的数据类型

```
var s:Sprite = new Sprite();
```

```
s is Sprite // true
```

```
s is DisplayObject // true
```

```
s is IEventDispatcher // true
```

```
s instanceof Sprite // true
```

```
s instanceof DisplayObject // true
```

```
s instanceof IEventDispatcher // false
```

# "::" 名称限定符

检查一个对象的命名空间

```
public namespace Dutch;  
public namespace French;
```

```
Dutch function hello():String {  
    return "hallo";  
}
```

```
French function hello():String {  
    return "bonjour";  
}
```

```
Dutch::hello()    // "hallo"
```

```
French::hello()   // "bonjour"
```

# "::" 名称限定符

"public", "private", "protected", "internal"  
这些也都是命名空间

```
public function get a():String;  
private function set a(value:String);
```

```
trace(a) // 编译错误  
a = "hello" // 编译错误
```

```
trace(public::a)  
private::a = "hello"
```

# "in" vs Object.hasOwnProperty

检查一个对象是否带有指定属性

<code>"CASEINSENSITIVE" in Array</code>	<code>// true</code>
<code>"CASEINSENSITIVE" in []</code>	<code>// false</code>
<code>"length" in Array</code>	<code>// true</code>
<code>"length" in []</code>	<code>// true</code>
<code>[] .hasOwnProperty("CASEINSENSITIVE")</code>	<code>// false</code>
<code>[] .hasOwnProperty("length")</code>	<code>// true</code>

# "arguments"

一个带有所有传递进来的参数的数组，任何方法中都可用

```
function myFunction (x:int) {  
    for(var i:uint=0; i<arguments.length; i++){  
        trace(arguments[i]);  
    }  
}  
myFunction(1, 2, 3);  
// 1  
// 2  
// 3
```



# "..." rest 参数

表示接收任意数量的参数

```
function myFunction (x:int, ... rest) {  
    for (var i:uint = 0; i< rest.length; i++) {  
        trace(rest[i]);  
    }  
}
```

```
myFunction(1, 2, 3);
```

```
// 2
```

```
// 3
```

# 提示与技巧

# 创建对象

```
var a:Array = new Array();
```

```
var a:Array = []; //这样更快
```

```
var o:Object = new Object();
```


```
var o:Object = {}; //这样更快
```

```
var v:Vector.<String> = new Vector.<String>();
```

```
v.push("a");
```

```
v.push("b");
```

```
var v:Vector.<String> = new <String>["a", "b"];
```



# 引用对象

```
var a:Object = {};  
a.name = "John";
```

```
var b:Object = a;  
b.name = "Elvis";
```

```
trace(a.name);           // 输出为 "Elvis"
```

# 对象复制

基于场景创建深表复制和浅表复制

// 深表复制

```
private function clone(obj:Object):Object {  
    var bytes:ByteArray = new ByteArray();  
    bytes.writeObject(obj);  
    bytes.position = 0;  
    return bytes.readObject();  
}
```

# 事件

始终覆写事件子类的"clone"方法，以防止重调时发生强制转换错误

```
class MyEvent extends Event {  
    public function MyEvent(data:Object) {  
        _data = data;  
    }  
    override public function clone():Event {  
        return new MyEvent(_data);  
    }  
}
```

# for...in vs. for each...in

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

```
// 基于键[0]、[1]、[2]的循环
```

```
for ( var i in arr ) {  
    trace( i );  
}
```

```
// 基于值"a"、"b"、"c"的循环
```

```
for each ( var s:String in arr ) {  
    trace( s );  
}
```

懂了吧：顺序不靠谱，结合计数器使用for循环

# trace()

trace()可以接受多个参数，没必要硬凑成一个字符串去trace

```
trace(new Date(2012, 4, 22), "Aloha", Math.PI, true);
```

```
// Tue May 22 00:00:00 GMT+0200 2012 Aloha 3.141592653589793  
true
```



# 循环标签

给循环贴上标签命名，在需要从嵌套循环中跳出时很有用

```
mainLoop:
```

```
for (var i:uint = 0; i<10; i++) {  
    for (var j:uint = 0; j<10; j++) {  
        if (i == 5 && j == 7) {  
            break mainLoop;  
        }  
    }  
}
```

```
}
```

# 全局函数

新建一个as文件，只申明一个与文件名相同命名的函数

// 文件名：myGlobalFunction.as

```
package {  
    function myGlobalFunction():void {  
        trace("in myGlobalFunction");  
    }  
}
```

# 为内建类添加方法

通过使用prototype在继承内建类特性的同时加入新方法

```
Array.prototype.removeItem = function (item:*) :void {  
    var index:int = this.indexOf(item);  
    if (index > -1) {  
        this.splice(index, 1);  
    }  
};
```

```
var a:Array = [1, 2, 3];  
a.removeItem(2);  
trace(a); // 1, 3
```

“ 懂了！ ”

&

“ 我X！ ”

# 显式转换

```
var o:MyObject = new MyObject();  
var o1:MyObject = MyObject(o);  
var o2:MyObject = o as MyObject; // o1 === o2
```

```
var a:Array = [1, 2, 3];  
var a1:Array = Array(a); // 新数组！！  
var a2:Array = a as Array; // a1 !== a2
```

操作Date和Error类时也适用，要注意！

# Boolean的显式转换法

```
Boolean(true)           // true
Boolean(false)          // false
Boolean(0)               // false
Boolean(1)               // true
Boolean(-1)              // true
Boolean("true")          // true
Boolean("false")         // true
Boolean("")              // false
Boolean(" ")             // true
Boolean("0")             // true
Boolean("1")             // true
Boolean(null)            // false
Boolean(undefined)       // false
Boolean(Object)          // true
Boolean({})              // true
```

# Array 类

```
var a:Array = new Array();
```

// 空数组

```
var a:Array = [];
```

// 空数组

```
var a:Array = new Array(10);
```

// 长度为10的数组

```
var a:Array = [10];
```

// 带有1个元素10的数组

```
var a:Array = new Array(1, 2, 3);
```

// 带有1、2、3三个值的数组

```
var a:Array = [1, 2, 3];
```

// 带有1、2、3三个值的数组

# Date 类

```
new Date(); // 当前日期
new Date(2012); // 01/01/1970 01:00:02
new Date(2012, 1); // 01/02/2012 00:00:00
new Date(2012, 1, 1); // 01/02/2012 00:00:00

public function Date(
    yearOrTimevalue:Object,
    month:Number, // 0 to 11
    date:Number = 1, // 1 to 31
    hour:Number = 0, // 0 to 23
    minute:Number = 0, // 0 to 59
    second:Number = 0, // 0 to 59
    millisecond:Number = 0) // 0 to 999
```



# 抛出

除了错误还有其他东西可以抛出

```
class AwesomeParty {}  
  
try {  
    throw new AwesomeParty();  
} catch (party:AwesomeParty) {  
    // go loose at moNo! (鬼知道啥意思)  
}
```

实用不？

# More info

## ActionScript 3 Language Reference

[http://help.adobe.com/en\\_US/FlashPlatform/reference/actionscript/3/](http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/)

## ActionScript 3 Language Specification

<http://livedocs.adobe.com/specs/actionscript/3/>

## Twitter

[@herrodus](#), [@stackandheap](#)

## Stack & Heap Labs

<http://labs.stackandheap.com>



# Questions ?

**Thank you !**