第5讲

Javascript入门

信息学院 孙辉

内容

- 认识Javascript
- JavaScript插入网页
- JavaScript语法

认识JavaScript

- 什么是JavaScript
- Java vs. JavaScript
- JavaScript可以做什么？

什么是JavaScript

- JavaScript最初是为了在HTML页面中增加交互功能而设计的
- JavaScript是一种脚本语言(脚本语言是一种轻型编程语言)
- 一个JavaScript脚本可包含多行可执行的计算机代码
- JavaScript脚本通常都直接嵌入在HTML页面里，也有时是写在单独的文件里
- JavaScript是一种解释型语言(这就意味着脚本执行时不需要预编译)

Java & JavaScript

- Java 和 JavaScript 无论从概念还是设计上来说都是两种完全不同的语言!

- Java (由Sun Microsystems公司开发) 是一种强大和更为复杂的编程语言 – 与C 和 C++属于同一级别

JavaScript能做什么

- JavaScript 为 HTML 设计者提供了一个编程工具
- JavaScript 可以对事件进行反应
  - 一段JavaScript 可以被设置为在某事件发生时执行，例如当一个页面完成加载时，或者当用户点击某个HTML控件时执行
- JavaScript可以读写HTML元素
- JavaScript 可以用来验证数据
- JavaScript 可以用来生成cookie

实例总体认识

```
<!DOCTYPE html>
<html>
<body>
    <h1>简单JavaScript实例</h1>
     <p>
          利用弹出对话框提示用户
      </p>
<script>
      document.write("Hello World!")
       alert("熊出没，注意");
</script>
</body>
</html>
```

效果1

```
<!DOCTYPE html>
<html>
<body>
<h1>我的第一段 JavaScript</h1>
<p id="demo">
JavaScript 能改变 HTML 元素的内容。
</p>
<script>
function myFunction()
{
      x=document.getElementById("demo");
       x.innerHTML="Hello JavaScript!";  // 改变内容
}
</script>
     <button type="button"
onclick="myFunction()">点击这里</button>
</body>
</html>
```

效果2

Another example 1

Another example 2

内容

- 认识Javascript
- JavaScript插入网页
- JavaScript语法

网页中插入JavaScript

- <script>标签
  - <script type="text/javascript"> </script> 旧方式
  - <script> </script> 新方式
  - 可以放在外部，javascript文件格式为*.js
    - <script src="myScript.js"></script>

- 插入的位置
  - 位置不同，效果不同
  - <head> </head> 之间， 一般为函数
    - 被调用的时候才会执行
  - <body> </body> 之间， 一般为过程执行
    - 在网页读取到该语句的时候就会执行

内容

- 认识Javascript
- JavaScript插入网页
- JavaScript语法

JavaScript的语句

- 语句

```
<script>
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
</script>
```

- 语句块

```
<script>
function myFunction()
{
document.getElementById("demo").innerHTML="My First JavaScript
Function";
}
```

JavaScript语法

- 基本要素
  - 区分大小写、注释、代码块
- 数据类型
- 运算符与表达式
- 变量
- 控制语句
- 函数

基本要素

- 区分大小写
- 注释
  - 同C/C++
  - //
  - /*　*/
- 代码块
  - {　}

词法结构

- 标识符
  - 必须以字母、_或者$开始，后续的字符可以是字母、数字、下划线或者美元符
- 保留字
- 可选的分号
  - ；语句分隔符
  - 如果语句独自占一行，可省略分号
    - a = 3;　　//可省略分号
    - b = 4;　　//可省略分号
    - a = 3; b = 4;

JavaScript表达式

- 把运算符和运算对象(操作数)连接起来，符合规则的JavaScript语法的式子，称JavaScript表达式

- 例如：

```
5+6  5-6
```

  - 5，6称为操作数(运算对象)
  - +，-称为运算符
  - 5+6，5-6都称为表达式

- 每个表达式都会返回一个值，比如5+6返回11，5>6返回false。

算数运算符

- 给定 y=5，下面的表格解释了这些算术运算符

运算符	描述	例子	结果
+	加	x=y+2	x=7
-	减	x=y-2	x=3
*	乘	x=y*2	x=10
/	除	x=y/2	x=2.5
%	求余数（保留整数）	x=y%2	x=1
++	累加	x=++y	x=6
--	递减	x=--y	x=4

- 用于字符串的 + 运算符

赋值运算符

- 给定 **x=10** 和 **y=5**，下面的表格解释了赋值运算符：

运算符	例子	等价于	结果
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

比较运算符

给定 x=5，下面的表格解释了比较运算符：

运算符	描述	例子
==	等于	x==8 为 false
===	全等（值和类型）	x===5 为 true；x==="5" 为 false
!=	不等于	x!=8 为 true
>	大于	x>8 为 false
<	小于	x<8 为 true
>=	大于或等于	x>=8 为 false
<=	小于或等于	x<=8 为 true

逻辑运算符

给定 x=6 以及 y=3，下表解释了逻辑运算符：

运算符	描述	例子
&&	and	(x < 10 && y > 1) 为 true
||	or	(x==5 || y==5) 为 false
!	not	!(x==y) 为 true

数据类型

- 字符串
- 数字
- 布尔
- Null
- Undefined

原始类型

- Javascript是一种基于对象的语言，数据类型本身可以定义方法（method)来使用值
- 例如：对数组a的元素排序，不需要将a传入sort()函数，而是调用a的一个方法　a.sort() // sort(a)的面相对象版本

- 数组
- 对象
  - 对象是属性的集合，每个属性都有“名/值对”

数据类型——文本

- 字符串
- 转义字符——\
- 字符串的使用
  - + 运算符：表示字符串连接
  - length：得到字符串的长度， s.length
  - 字符串还是提供了其他很多可以调用的方法
    - s.substring(1, 4)
- 模式匹配
  - RegExp()构造函数，用来创建文本模式匹配的对象
    - 模式称为“正则表达式”（perl语言的正则表达式语法）
    - String和RegExp对象都定义了利用正则表达式进行模式匹配和查找与替换的函数

```
var  text = “testing: 1, 2, 3”;
var pattern = /\d+/g //匹配一个或多个数字的实例
pattern.test(text)  // true，匹配成功
text.search(pattern)// 9, 首次成功匹配的位置
test.match(pattern)//[“1”, “2”, “3”]：所有匹配组成的数组
```

数据类型——数字

- Javascript不区分整数值和浮点数值
  - 统一用64位浮点格式表示数字
  - 注意：最大最小值、表示精度
    - 精度： 精度不准确，小数最大位数17位。
  - 1和1.0是一样的
- 算数运算
  - + - * / %(求整除后的余数)
  - 基本运算符外，支持复杂的算数运算，通过作为Math对象的属性定义的函数和常量来实现
    - Math.pow(2,53)
    - Math.PI
    - Math.sqrt(3)
- 日期和时间
  - Date()构造函数　　var then = new Date(2014, 0, 1, 17, 10, 30)

数据类型（3）

- 布尔值

- null和undefined
  - null：特殊值，表示“空值”
  - undefined：表示更深层次的“空值”

- 全局对象
  - 全局属性：undefined, NaN
  - 全局函数：isNaN()
  - 构造函数：Date(), RegExp()
  - 全局对象：Math, JSON

变量

- 变量是存储信息的容器
- 举例：

```
var x=2;
var y=3;
var z=x+y;
```

- 变量名必须以字母、$、_开头
  - 不推荐以$和_符号开头
- 变量名称对大小写敏感（y和Y是不同的变量）
- 弱类型语言
  - 无需声明，直接使用

提示：一个好的编程习惯是，在代码开始处，统一对需要的变量进行声明。

字符串变量

- JavaScript拥有动态类型，相同的变量可用作不同的类型

```
var x                // x 为 undefined
var x = 6;           // x 为数字
var x = "Bill";      // x 为字符串
```

- JavaScript 字符串引号中的任意文本，可以使用单引号或双引号

```
var carname="Bill Gates";
var carname='Bill Gates';
```

实例

数字变量

- JavaScript 只有一种数字类型。数字可以带小数点，也可以不带

```
var x1=34.00;      //使用小数点来写
var x2=34;         //不使用小数点来写
```

- 极大或极小的数字可以通过科学（指数）计数法来书写

```
var y=123e5; // 12300000
var z=123e-5; // 0.00123
```

数组变量

- JavaScript 数组
  - 数组下标是从0开始
  - 关联数组

```
var cars=new Array();
cars[0]="Audi";
cars[1]="BMW";
cars[2]="Volvo";
```

或：

```
var cars=new
Array("Audi","BMW","Volvo");
```

或：

```
var cars=["Audi","BMW","Volvo"];
```

```
var postcodes=[];
postcodes["canada"] = 012;
postcodes["china"]=086;
```

对象变量

- **JavaScript 对象**
  - 对象由花括号分隔。在括号内部，对象的属性以名称和值对的形式 (name : value) 来定义。属性由逗号分隔

```
var person={firstname:"Bill", lastname:"Gates", id:5566};
```

  - person有3个属性： firstname、lastname 以及 id。
  - 对象属性有两种寻址方式：

```
name=person.lastname;
name=person["lastname"];
```

对象变量

- **JavaScript 对象**
  - 对象由花括号分隔。在括号内部，对象的属性以名称和值对的形式 (name : value) 来定义。属性由逗号分隔

```
var person={firstname:"Bill", lastname:"Gates", id:5566};
```

  - person有3个属性： firstname、lastname 以及 id。
  - 对象属性有两种寻址方式：

```
name=person.lastname;
name=person["lastname"];
```

声明变量类型

- 当您声明新变量时，可以使用关键词 "new" 来声明其类型

```
var carname =    new String;
var x=           new Number;
var y=           new Boolean;
var cars=        new Array;
var person=      new Object;
```

JavaScript 变量均为对象。当您声明一个变量时，就创建了一个新的对象。

JavaScript 对象

- JavaScript 中的所有事物都是对象：字符串、数字、数组、日期……
- JavaScript 中，对象是拥有属性和方法的数据
- 属性和方法
  - 属性是与对象相关的值。
  - 方法是能够在对象上执行的动作。
  - 例如：汽车——现实生活中的对象
    - 属性：　car.name=Fiat　　car.model=500
      car.weight=850kg　car.color=white
    - 方法：car.start()　　car.drive()　　car.brake()

JavaScript 对象（2）

- var txt = "Hello";
- 实际上已经创建了一个 JavaScript 字符串对象
- 字符串对象拥有内建的属性 length
- 属性：　txt.length=5
- 方法：　txt.indexOf()
  txt.replace()
  txt.search()
- 访问对象的属性：*objectName.propertyName*
  访问对象的方法：*objectName.methodName()*

字符串对象的方法

bold()	粗体
italtics()	斜体
strike()	删除线
fontsize(字级大小)	文字大小
fontcolor(#rrggbb)	文字颜色
sup()	上标
sub()	下标
toUpperCase()	大写
toLowerCase()	小写
charAt(索引)	返回索引位置的字符
charCodeAt(索引)	返回索引位置的ASCII字符码，十进制表示
indexOf("字串"[,索引])	返回字串在对象中的索引位置
lastIndexOf("字串"[,索引])	返回字串在对象中的索引位置（反向搜索）
search("字串")	返回字串在对象中的索引位置
replace("字串1","字串2")	字串2替换字串1
slice(索引i[,索引j])	返回索引i倒索引j-1的子串
split(["字串"][,限制])	将字串从对象中删除
substr(start[,length])	返回特定长度的字串
substring(索引i[,索引j])	返回索引i倒索引j-1的子串
link("url")	设置链接
match(/匹配字符/)	\d 匹配一个数字字符。 \D 匹配一个非数字字符。 \n 匹配一个换行符。 \r 匹配一个回车符。 \s 匹配一个空格符。 \S 匹配任意非空格符。 \t 匹配一个table符。 \W 匹配任何非单词符。 \num 匹配正整数num。 /n/ 匹配八进制，十六进制，十进制的escape值。
toString()	返回字符串
valueOf()	返回字符串值

数学对象的属性&方法

属性

E	自然对数的底数
LN2	2的自然对数
LN10	10的自然对数
LOG2E	以2为底e的对数
LOG10E	以10为底e的对数
PI	圆周率
SQRT1_2	1/2的平方根
SQRT2	2的平方根

方法

ceil(数值)	大于等于该数值的最小整数
floor(数值)	小于等于该数值的最大整数
min(数值1,数值2)	最小值
max(数值1,数值2)	最大值
pow(数值1,数值2)	数值1的数值2次方
random()	0到1的随机数
round(数值)	最接近该数值的整数
sqrt(数值)	开平方根

abs、sin(弧度)、cos、tan、asin、acos、atan、exp、log

控制程序流

- 条件，循环　类似C
  - If,else, while, do while, for

  - 特别的：
    - for/in　　循环遍历对象的属性
      - var person={fname:"John",lname:"Doe",age:25};
      - var person2 = [“first”,”second”];
      - for (x in person) { txt=txt + person[x]; }
      - for (x in person2) { txt=txt + person[x]; }

条件语句

- if 语句
  - 只有当指定条件为 true 时，使用该语句来执行代码
- if...else 语句
  - 当条件为 true 时执行代码，当条件为 false 时执行其他代码
- if...else if....else 语句
  - 使用该语句来选择多个代码块之一来执行
- switch 语句
  - 使用该语句来选择多个代码块之一来执行

实例

循环语句——for

- 循环可以将代码块执行指定的次数

```
document.write(cars[0] + "<br>");
document.write(cars[1] + "<br>");
document.write(cars[2] + "<br>");
document.write(cars[3] + "<br>");
document.write(cars[4] + "<br>");
document.write(cars[5] + "<br>");
```

```
for (var i=0;i<cars.length;i++)
{
        document.write(cars[i] + "<br>");
}
```

循环语句

- JavaScript 支持不同类型的循环：
  - **for** - 循环代码块一定的次数
  - **for/in** - 循环遍历对象的属性
  - **while** - 当指定的条件为 true 时循环指定的代码块
  - **do/while** - 同样当指定的条件为 true 时循环指定的代码块

JavaScript for/in循环

- 循环遍历对象的属性

```
var person={fname:"John",lname:"Doe",age:25};

for (x in person)
{
        txt=txt + person[x];
 }
```

实例

break语句&Continue语句

- break 语句用于跳出循环
- continue 用于跳过循环中的一个迭代

函数

- 函数定义：函数是由事件驱动的或者当它被调用时执行的可重复使用的代码块。
- 基本定义方法： 类似C 命名方式或声明式

```
function functionname(xxx，yyy){
  这里是要执行的代码
  return XXX;
}
```

  - 调用方式

```
<button onclick="myFunction('Bob','Builder')">点击这里</button>
```

  - 函数内部变量 生存周期
    - 有声明
    - 无声明
  - 如果您把值赋给尚未声明的变量，该变量将被自动作为全局变量声明。

局部变量&全局变量

- 函数内部声明的变量（使用 var）是**局部**变量
  - 只能在函数内部访问它
  - 变量的作用域是局部的
  - 只要函数运行完毕，本地变量就会被删除

- 在函数外声明的变量是**全局**变量，网页上的所有脚本和函数都能访问它

- 赋值给尚未声明的变量，该变量将被自动作为全局变量声明。

JavaScript 计时事件

- 用以指定在一段特定的时间后执行某段程序，而不是在函数被调用后立即执行
  - setTimeout()：未来的某时执行代码
  - clearTimeout()：取消setTimeout()

实例

setTimeout()

var t=setTimeout("javascript语句",毫秒)

- 第一个参数是含有 JavaScript 语句的字符串。这个语句可能诸如 "alert('5 seconds!')"，或者对函数的调用，诸如 alertMsg()“
- 第二个参数指示从当前起多少毫秒后执行第一个参数。
- 如果希望取消这个 setTimeout()，可以变量名 t来指定它

clearTimeout()

JavaScript 错误处理

- try 语句测试代码块的错误
- catch 语句处理错误

语法

```
try
  {
  //在这里运行代码
  }
catch(err)
  {
  //在这里处理错误
  }
```

实例1

- throw 语句创建自定义错误
  - throw exception
  - 异常可以是 JavaScript 字符串、数字、逻辑值或对象

实例2

事件

- 用户对浏览器所做的特定的动作（操作），是实现交互操作的一种机制

事件名称	适用对象	意义	说明
Abort	image	终止	当图形尚未完全加载前
Blur		失去焦点	
Change	t/pw/ta/select	改变	
DragDrop	window	拖曳	
Error		img/win	错误加载文件或图形时发生错误
Focus		取得焦点	
Move	window	移动	
Reset	form	重置	
Submit	form		
Click/DblClick、KeyDown/KeyPress/KeyUp、Load/Unload、MouseDown/MouseUp/MouseOver/MouseOut/MouseMove			

作业：进一步自学JavaScript

- http://www.codecademy.com/tracks/javascript