Javascript

讲师：陈伟俊

概述及特点

- 是一种脚本语言，在程序运行中逐行解释，而不需要实现编译。
- 基于对象，能够使用存在的程序对象，但它无法像Java一样完全面向对象。
- 掌握比较容易，不需要太多的编程基础。
- 能够跨平台，JavaScript主要是通过浏览器进行解释执行，和操作系统不发生直接关系，能够运行浏览器软件的地方就可以运行JavaScript，例如微软的IE和网景的NN这两个主流浏览器都支持JavaScript。
- 通过浏览器执行，根据浏览器不同执行可能会有差异

第一个javascript程序

- 在网页中加入以下代码

```
<script type="text/javascript">
   document.write("hello,world");
</script>
```

数据类型和变量

JavaScript的基本数据类型可以分成4种

- 数字型：包括整型和浮点型，例如2，5，3.14
- 字符串型：需要放到双引号中，例如”hello”，”中国”
- 布尔型：只有两个取值，true（真）和false（假）
- 空值：null表示什么都没有

使用变量

- JavaScript是采用弱数据类型的语言，意味着声明变量时不需要指定变量的数据类型，数据在被赋给变量的时候会自动确定数据类型
- 在JavaScript中的声明一个变量的语法如下：
  var 变量名;
- 也可以用一条语句声明多个变量：
  var 变量名1,变量名2,变量名N;
- 以下代码是合法的变量声明：
  var a1; 或者 var a1 = 10;
  var a2,a3; 或者 var a2,a3 = 10;

注释

- 单行注释 //
- 多行注释/*　*/

算术运算符

- + 加法运算，例如5+3的结果为8
- - 减法运算，例如5-3的结果为2
- * 乘法运算，例如5*3的结果为15
- / 除法运算，例如5/2的结果为2.5
- % 求模运算 例如5%3的结果为2
- ++ 递增运算，变量自身加1，例如var i = 10;i++的结果为11
- -- 递减运算，变量自身减1，例如var i = 10;i--的结果为9

比较运算符

- == 判断两边的值是否相等
- > 判断左边的值是否大于右边的值
- < 判断左边的值是否小于右边的值
- >= 判断左边的值是否大于等于右边的值
- <= 判断左边的值是否小于等于右边的值
- != 判断两边的值是否不相等

逻辑运算符

- && 逻辑与，左右两边的表达式都为true的时候返回true
- || 逻辑或，左右两边的表达式有一个为true的时候返回true
- ! 逻辑非，表达式为true返回false，表达式为false返回true

赋值运算符

- = 将右边的值赋给左边的变量，例如 i=6
- += 例如i+=6等价于i=i+6
- -= 例如i-=6等价于i=i-6
- *= 例如i*=6等价于i=i*6
- /= 例如i/=6等价于i=i/6

If语句

```
if ( <条件判断> ){
    <执行代码>
}
```

if…else…语句

```
if ( <条件判断> ){
    <执行代码>
}else {
    <执行代码>
}
```

if…else if…else…语句

```
if ( <条件判断> ){
    <执行代码>
} else if ( <条件判断> ){
    <执行代码>
} else if ( <条件判断> ){
    <执行代码>
}
……
else {
    <执行代码>
}
```

switch语句

```
switch (<表达式>){
    case <值1>：
            <代码>
            [break;]
    case <值2>：
            <代码>
            [break;]
    ……
    [
    default ：
            <代码>
    ]
}
```

for循环语句

```
for (<变量的初始化>;<条件表达式>;<迭代>){
    <代码>
}
```

示例：

```
<SCRIPT language="javascript">
    for (var i = 1;i <= 10;i ++){
            document.write(i + " ");
    }
</SCRIPT>
```

while循环语句

```
while (<逻辑表达式>){
    <代码>
}
```

示例

```
var i = 1;
while (i <= 10){
    document.write(i + " ");
    i ++;
}
```

自定义函数

自定义函数类似于java中的方法，将程序模块化，可以被重复调用。

```
function 自定义函数名([参数列表]){
   <代码>
   [return 返回值]
}
```

自定义函数示例

```
function getResult(a,b){
   return a+b;
}

alert(getResult(5,3));
alert(getResult(10,20));
```

事件处理

- 事件是浏览器响应用户的操作的机制，“事件驱动”是JavaScript程序的特点。例如：一个用户点击了一个按钮，即一个用户对按钮进行了单击操作，产生了单击事件，告诉浏览器需要处理，这时候浏览器响应这个事件，并执行相应的事件处理程序。

事件处理

```
<HTML>
  <HEAD>
      <TITLE>一个简单的事件处理</TITLE>
  </HEAD>
  <BODY>
      <INPUT type="button" value="click here"
                  onClick="alert('hello,world!')">
  </BODY>
</HTML>
```

网页元素对应的事件名称

网页元素	事件名称	说明
链接	onClick	单击链接
	onMouseOver	鼠标移上链接
	onMouseOut	鼠标移出链接
图像	onAbort	用户中断图像载入
	onError	图像载入时发生错误
	onLoad	图像已经载入并显式
网页主体	onBlur	失去焦点
	onError	载入网页发生错误
	onFocus	获得焦点
	onLoad	网页载入
	onUnLoad	网页关闭

网页元素对应的事件名称

表单	onSubmit	提交表单
	onReset	重置表单
文本框	onBlur	文本框失去焦点
	onFocus	文本框获得焦点
	onSelect	文本框文字被选择
按钮	onClick	单击按钮
单选框	onClick	选择单选框
复选框	onClick	选择复选框
列表框	onSelect	选择列表框中项目

JavaScript内置对象

- JavaScript语言与Java不同，并不是完全面向对象，而是基于对象的，它可以创建一些内置对象的实例，然后调用这些对象的属性和方法。
- JavaScript提供了关于日期时间，字符串还有数字方面的内置对象。

Date对象

- Date对象提供了对于日期时间处理的一系列方法

方法名	说明
getDate()	返回天数
getDay()	返回星期几
getHours()	返回小时数
getMinutes()	返回分钟数
getMonth()	返回月份
getSeconds()	返回秒数
getTime()	返回时间（毫秒单位）
getYear()	返回年份

Date对象

构造方法名	说明
Date()	用当前时间创建实例
Date(年,月,日)	用指定的年月日创建实例
Date(年,月,日,时,分,秒)	用指定的年月日时分秒创建实例
Date(“月/日/年”)	用指定的日期创建实例

Date对象示例

```
date = new Date();

document.write(date.getYear() + "年" + (date.getMonth()+1)
 + "月" + date.getDate() + "日");

document.write(date.getHours() + ":" + (date.getMinutes()+1)
 + ":" + date.getSeconds());
```

String对象

方法名	说明
charAt(索引数字)	返回指定索引位置的字符
indexOf(字符串)	返回参数字符串在String对象字符串中第一次出现的索引位置，如果没有返回-1
indexOf(字符串,起始位置)	同上，只是指定查找参数字符串时的起始位置
lastIndexOf(字符串)	返回参数字符串在String对象字符串中最后一次出现的索引位置，如果没有返回-1
lastIndexOf(字符串,起始位置)	同上，只是指定查找参数字符串时的起始位置
split(分隔符)	将字符串按指定分隔符分解成一个字符串数组
substring(起始位置，终止位置)	返回从起始位置到终止位置之间的字符串
toLowerCase()	返回转换为小写的字符串
toUpperCase()	返回转换为大写的字符串

String对象

- String对象还有一个length属性，返回字符串的长度

Math对象

方法名	说明
abs(数字)	返回绝对值
ceil(数字)	进位取整
floor(数字)	返回不大于本身的最大整数
max(数字x,数字y)	返回最大值
min(数字x,数字y)	返回最小值
pow(数字x,数字y)	返回x的y次方
random()	返回0～1之间的随机数
round(数字)	四舍五入取整
sqrt(数字)	返回平方根

数组对象

- 数组也是JavaScript中的一种对象，使用的时候和其他对象一样，需要创建一个数组对象的实例，例如下面的代码就是创建了一个有5个元素的数组对象：

```
var a = new Array(5);
a[0] = 10;
a[1] = 20;
a[2] = 30;
a[3] = 40;
a[4] = 50;
```

数组对象

- 创建数组对象其他方法
  - a = new Array(10,20,30,40,50);
  - a = [10,20,30,40,50];
- 数组长度：length属性

内置函数eval

- eval(字符串)
- 接收一个字符串形式的表达式，并试图求出表达式的值。作为参数的表达式可以采用任何合法的操作符和常数。
- 如果传递给这个函数的参数中包含JavaScript命令，这些命令也可以被执行，就像这些命令是JavaScript程序的一部分一样。

内置函数parseInt

- parseInt (字符串)
- 试图从一个字符串中提取一个整数。parseInt函数也可以附加一个整数n，可以返回n进制的一个整数。如果在字符串中存在除了数字、符号、小数点和指数符号以外的字符，parseInt函数就停止转换，返回已有的结果。如果第一个字符就不能转换，函数就返回“NaN”值。

内置函数parseFloat

- parseFloat (字符串)
- 函数试图从一个字符串中提取一个浮点值。如果在字符串中存在除了数字、符号、小数点和指数符号以外的字符，parseFloat函数就停止转换，返回已有的结果。如果第一个字符就不能转换，函数就返回“NaN”值。

内置函数isNaN

- 当JavaScript遇到一个使用parseint函数和parsefloat函数中的任何一个都不能转换成数字的字符串时，将自动返回一个叫做NaN的结果。isNaN函数可以测试这两个函数返回的结果是否为NaN，如果是，函数返回true。

DOM（Document Object Model）文档对象模型

- 每一个网页元素都是一个对象，例如窗口，文档页面，图片，图层，表格都是一个对象，如果元素标签还有嵌套标签，还有子对象的概念

对象模型

navigator浏览器对象
screen屏幕对象
window窗口对象

- history历史对象
- location地址对象
- frames[]; Frame框架对象
- document文档对象
  - anchors[]; links[]; link连接对象
  - forms[]; Form表单对象
    - Button按钮对象
    - Checkbox复选框对象
    - elements[]; Element表单元素对象
    - Hidden隐藏对象
    - Password密码输入区对象
    - Radio单选域对象
    - Reset重置按钮对象
    - Select下拉菜单、列表对象
    - options[]; Option选择项对象
    - Submit提交按钮对象
    - Text文本框对象
    - Textarea多行文本输入区对象
  - images[]; Image图片对象

得到对象的引用常见方法

- 父对象.子对象.子子对象
  - 例如window.document.form1.t1
- document.getElementById(“对象ID”)
  - 例如 document.getElementById(“div1”);

常用DOM对象的属性方法

- 打开新窗口
  - window.open(“url”,”窗口名字”,”窗口设置”)
- 弹出对话框
  - window.alert(内容);
- 弹出确认框，确定返回true 取消返回 false
  - window.confirm(内容)
- 隔一定毫秒数执行一次指定语句
  - window.setTimeout(“语句”,毫秒数)

常用DOM对象的属性方法

- 每隔一定毫秒数执行指定语句
  - window.setInterval(“语句”,毫秒数)
- 获得网页元素的引用
  - document.getElementById(“元素ID名”)
- 跳转到其他URL
  - location.href=url地址
- 后退上一页
  - history.back()
- 设置或返回标签对象中的HTML内容的属性
  - 对象.innerHTML

表单操作

获得表单对象的三种方法

- document.forms[下标]
- document.forms[“表单名”]
- document.表单名

表单对象的常见属性和方法

- 属性
  - elements 获得表单种所有控件的集合
  - length 表单元素的个数
- 方法
  - submit() 提交表单
  - reset() 重置表单

表单元素(表单域)的获取三种方法

- 表单对象.elements[下标]
- 表单对象.elements[“表单元素名称”]
- 表单对象.表单元素名称

表单元素的通用属性和方法

- 属性
  - disabled 元素失效 true或false
  - name 元素名称
  - value 元素的值
- 方法
  - focus() 获得焦点

单选按钮和复选按钮

- radio和checkbox对象均返回一个数组，可以通过下标去访问其中具体的表单元素
- 常见属性
  - length元素的个数
  - checked 元素是否被选中

列表框

- 常用属性
  - length option的个数
  - selectedIndex 当前选中的索引，没有选择任何元素返回-1
  - options 所有options的数组集合

表单验证

- 在<form>中设置
  - onsubmit＝return 验证函数;
  - 根据验证函数的返回值决定提交是成功还是失败

正则表达式

- 正则表达式主要利用通配符做模糊匹配，可以有效的判断字符串格式。
- 创建一个正则表达式对象：
  - var 变量 = /模式/[gi]
  - var 变量 = new RegExp(“模式” [, “gi”]);
- g表示全局匹配用在replace中，i表示忽略大小写
- 判断字符串是否匹配当前的正则表达式模式方法
  - 正则表达式对象.test（字符串） 返回true或false

正则表达式举例

```
var r = /ab/; //正则表达式对象
var str = “abc”; //字符串

document.write(r.test(str));//true

r = /AB/;
document.write(r.test(str));//false

r = /AB/i;
document.write(r.test(str));//true
```

正则表达式元字符

- \ 转义字符
- ^ 表示开始 例如 /a/ 匹配 bab 但 /^a/ 匹配 ab 不匹配bab
- $ 表示结束

表示某些字符的元字符

- . 表示任意字符 包括中文
- [abc] 表示a或b或c其中一个字符
- [^abc] 必须不是其中任何一个字符
- [a-z] 小写字符 [A-Z]大写字母 [a-zA-Z]所有字母 也可以是[c-e][X-Z]
- [0-9] 数字
- [a-zA-Z0-9] 字母和数字
- [^a-z] [^A-Z][^0-9] 非字母，非数字
- \s 空白（空格，制表位，换行，换页，回车）
- \S 非空白
- \d 数字 相当于[0-9]
- \D 非数字 相当于[^0-9]
- \w 字母或数字 [0-9][a-z][A-Z]
- \W 非字母或数字

表示匹配次数的元字符

- * 0次到多次
- + 1次到多次
- ？ 0次或1次
- {n} 真好n次
- {n,} 至少n次
- {n,m} n次到m次

其他元字符

- | 逻辑或 要打小括号
- (表达式) 子表达式,改变优先级

应用外部js文件

- 可以将**javascript**程序单独写在一个外部**.js**文件中，然后在另一个文件中调用
- 例如：
- **script.js**代码

```
function showTxt(){
   alert("外部javascript程序被调用");
}
```

应用外部js文件

- page.html代码

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>调用外部javascript</title>
<script type="text/javascript" src="script.js"></script>
</head>
<body onLoad="showTxt()">
<p>调用外部javascript程序</p>
</body>
</html>
```