**JavaScript**

## 1 JavaScript概念

JavaScript是一种面向对象、基于事件、解释型、脚本语言。

### 1.1 JavaScript特点

不是Java的子集。(livescript)

不是只能运行在浏览器端。

**弱类型语言**（定义变量时，不需要分配数据类型）。

默认使用的字符集和java一样是unicode(两个字节一个字符)。

和java一样，区分大小写。

和java一样，使用分号表示一句话的结束。

注释方式和java一样 // 单行注释 /\* \*/多行注释。

### 1.2 JavaScript版本

European Computer Manufacturer's Association

ECMAScript-262 Standard

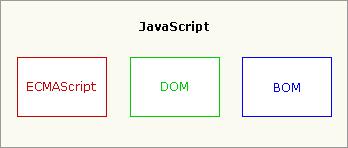
ECMAScript V3

ECMAScript 6

JavaScript1.6

**JavaScript2.0**

### 1.3 JavaScript的组成



* ECMAScript (核心)：规定了JS的**语法和基本对象**。

JavaScript 没有提供文档

微软：JScript 提供了特别详细的文档 淘汰

* DOM 文档对象模型：**处理网页内容**的方法和接口

标记型文档。HTML

document 文档

接口是由各大浏览器实现。

IE

火狐

谷歌

操作HTML/CSS展示特效

* BOM 浏览器对象模型：**与浏览器交互**的方法和接口

接口是由各大浏览器实现。

IE

火狐

谷歌

操作浏览器特效

### 1.4JavaScript的引入方式

#### 1.4.1内部脚本

在当前页面内部写script标签，script内部即可书写JavaScript代码

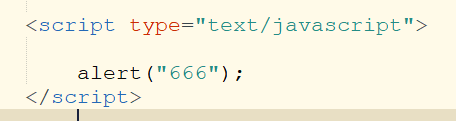
格式：

<script type=”text/javascript”>

JavaScript的代码

</script>

示例：



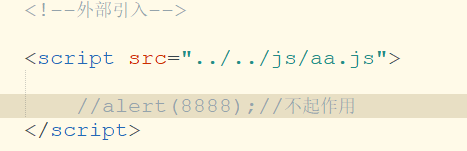
注：script标签理论上可以书写在HTML文件的任意位置

#### 1.4.2外部引入

在HTML文档中，通过<script src=””>标签引入.js文件

格式：

<script type=”text/javascript” **src**=”javascript文件路径” ></script>



### 1.5 JavaScript中的注释

单行注释 //

多行注释 /\* \*/

## 2 JavaScript的常量

12               // 整数类型常量

1.2              // 浮点数类型常量

"hello world"    // 字符串类型常量

'Hi'             // 字符串类型常量

true             // 布尔类型常量

false            // 布尔类型常量

/javascript/gi   // 正则表达式

null             // 空

**{ x:1, y:2 }** // javascript中的对象 json格式

[1,2,3,4,5]     // javascript中的数组

## 3 JavaScript中的数据类型

JavaScript 的变量定义：

**var 变量名 = 值;**

**变量名命名规则和Java语言类似：**

必须以字母或下划线开头，中间可以是数字、字符或下划线

变量名不能包含空格等符号

不能使用JavaScript关键字作为变量名，如：function、this、class

JavaScript严格区分大小写。 建议使用小写开头

|  |
| --- |
| Numbers  Strings  Booleans  Objects  null  undefined  Arrays |

### 3.1 Numbers（数字类型）

JavaScript中只有一种数字类型。64位浮点数。遵循IEEE-754标准，类似于Java中的double类型。

var x = 10； //整数

var x = 017 //八进制

var x = 0x16 //十六进制

var x = 3.14 //浮点数

var x = .33 //浮点数

var x = 6.02e23 //科学计数法

**JavaScript浮点数运算同样结果不精确**

**var a = 0.1;**

**var b = 0.2;**

**var result = a+b; //0.30000000000000004**

**数字类型函数：**

**1 Number(value)：将其他类型转换成数字类型**

Number("20") -------------20

Number("20ab22")--------NaN

Number("abc")------------NaN

+"20"----------------------20

**2 parseInt(value)、parseFloat(value)**

parseInt("3 blind mice");    // Returns 3  
 parseInt("12.34");           // Returns 12  
 parseInt("0xFF");            // Returns 255

parseInt("11", 2);           // Returns 3 (1\*2 + 1)  
parseInt("ff", 16);          // Returns 255 (15\*16 + 15)  
parseInt("zz", 36);          // Returns 1295 (35\*36 + 35)  
parseInt("077", 8);          // Returns 63 (7\*8 + 7)  
parseInt("077", 10);         // Returns 77 (7\*10 + 7)

parseInt("eleven");          // Returns NaN  
parseFloat("$72.47");        // Returns NaN

parseFloat("3.14 meters");   // Returns 3.14

**3 NaN (Not a Number)**

表示不是数字

NaN的特点:

NaN和任何东西都不等（NaN不等于NaN）

判断一个值是不是NaN，使用isNaN(value)

alert(isNaN(r1));

alert(isNaN(100));

**4 JavaScript中的Math**

abs 绝对值

floor 地板数

log 对数

max 最大值

random 随机数

round 四舍五入取整

sqrt 开平方

|  |
| --- |
| var r1 = Math.abs(-5);  //alert(r1);  var r2 = Math.random();  //alert(r2)  var r3 = Math.floor(4.7);  //alert(r3);  var r4 = Math.ceil(4.7);  //alert(r4);  var r5 = Math.round(4.1);  //alert(r5);  var r6 = Math.max(10001,100,200,300,4000);  alert(r6); |

±1.7976931348623157 x 10308 ±5 x 10-324.

**5 Number类型的常量**

Number.MAX\_VALUE

Number.MIN\_VALUE

Number.NaN

Number.POSITIVE\_INFINITY

Number.NEGATIVE\_INFINITY

Number. Infinity

### 3.2 String(字符串类型)

在JavaScript中没有字符类型。

JavaScript中的字符串可以使用双引号，也可以使用单引号。

在JavaScript中获取字符串的长度，使用length属性。

|  |
| --- |
| var s = "abcde";  var s2 = ‘中文’;  //alert(s);  //alert(s2);  alert(s.length);  alert(s2.length); |

**关于字符串的一些方法：**

charAt

concat

indexOf

lastIndexOf

match 匹配正则表达式

replace

split

var s = "a#bb#c#dd";

var ss = s.split('#');

for(var i =0;i<ss.length;i++){

alert(ss[i]);

}

substring

toLowerCase

toUpperCase

**字符串运算：**

var x = "21";

var y = "2";

var result = x+y; "212"

var result = x-y; 19

var result = x\*y; 42

var result = x/y 10.5

**toString() 方法**

var n = 17;  
binary\_string = n.toString(2);        // "10001"  
octal\_string = "0" + n.toString(8);   // "021"  
hex\_string = "0x" + n.toString(16);   // "0x11"

### 3.3 Boolean（布尔类型）

**布尔类型常量：** true false

JavaScript中，其他类型可以自动转型为布尔类型，规则如下：

Number 非0 为true 0 为false

String 非空串 为true 空串为false （"" ''）

null------------false

undefined------false

Object---------不为null--true

Arrays----------不为null—true

|  |
| --- |
| var n = 'aaa';  if(n){  alert("OKOK");  }  **for(var i=10;i--;){**  **alert(i);**  **}** |

### 3.4 function（函数）

Javascript中定义函数，使用**function**关键字。

|  |
| --- |
| **定义函数：**  function fun(){  alert(111);  }  **调用函数：**  fun()  **定义有参的函数：**  functionadd(x,y){  alert(x+y);  }  **调用有参函数：**  add(100,200)  **定义有返回值的函数:**  function add(x,y){  return x+y;  }  **调用有返回值的函数：**  var result = add(100,200);  **注意：js的函数在定义时没有返回值类型，在函数使用return语句，表示函数返回值。** |

JavaScript中函数：

1 有函数名的函数： function add(){}

2 没有函数名，有引用指向的函数：

var x = function(x,y){} 调用：x(100,200)

3 没有函数名，没有引用指向的函数（直接调用）：

(function(x){alert(x)})(4);

**示例：**

var result=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 求4的阶乘

|  |
| --- |
| (function (x){    var result = 1;  for(i=1;i<=x;i++){  result = result\*i;  }    return result;  })(4) |

### 3.5 Object(对象类型)

java基于模版的面向对象（要使用对象，要先创建类）

javascript是基于原型的面向对象

**Javascript中创建对象的三种方式：**

**方式一：原型法**

|  |
| --- |
| var user = new Object();  user.id=1;  user.name="zhangsan";  user.sayHello=function(){  alert("hello.......");  }  user.add=function(x,y){  return x+y;  }  **调用：**  **alert(user.id);**  **alert(user.name);**  **user.sayHello();**  **var result = user.add(1,2);**  **alert(result);** |

**方式二：构造函数法**

|  |
| --- |
| function User(id,name,age){  this.id=id;  this.name=name;  this.age=age;  this.sayHello=function(){alert("你好");}  this.add = function(x,y){return x+y;}  this.myMinus=minus; // 指向外部的一个函数  }  function minus(x,y){  return x-y;  }  **使用：**  var user = new User(1,'小张',20); //创建对象  alert(user.id);  alert(user.name);  alert(user.age);  user.sayHello();  alert(user.add(2,5));  alert(user.myMinus(100,98)); |

**方式三：常量法（最常见） JSON**

|  |
| --- |
| var user = {  id:1,  name:'张三',  age:20,  sayHello:function(){  alert("hello");  },  add:function(x,y){return x+y;},  myMinus:minus // 指向外部的函数  }  function minus(x,y){ return x-y}  **使用：**  alert(user.id+" "+user.name+" "+user.age);  user.sayHello();  alert(user.add(1,2));  alert(user.myMinus(1,2)) |

### 3.6 Array(数组类型)

java中的数组是静态数组。 String[] = new String[5];

JavaScript中的数组是动态数组。

**定义数组：**

var aa1 = [1,3,9,10];

var aa2 = [4,6,"abc","ccc",{x:1,y:2}]; 由于js是弱类型语言，所以数组中可以存任何类型数据。

**访问数组元素：**

alert(as[2]);

alert(as[3]);

as[4]=100; //超出范围赋值也可以

alert(as[4]);

**其他定义数组的方式：**

var arr = new Array(); 相当于 var arr = [];

arr[0]="aaa";

arr[1]='bbb';

arr[2]= 'ccc';

var arr = new Array('aaa','ccc','bbb');

var arr = new Array(10,11,68) // 由 10 11 68 组成的数组

var arr = new Array(10)//10个长度的数组，数组中还没有赋值

**JS中的数组排序：**

var aa = [1,5,3,14,111];

aa.sort();

aa.sort(function(a,b){ // 回调

return a-b;

});

**对象数组排序：**

|  |
| --- |
| var u1 = {id:5,name:'aaa'};  var u2 = {id:1,name:'ccc'};  var u3 = {id:10,name:'bbb'};  var us = [u1,u2,u3];  us.sort(function(a,b){  return a.id-b.id; // 按id排序  }); |

**按name排序：**

|  |
| --- |
| us.sort(function(a,b){  if(a.name>b.name){  return 1;  }  if(a.name==b.name){  return 0;  }  if(a.name<b.name){  return -1;  }  }); |

**数组的函数：**

1 sort()

sort(function(a,b){})

2 join() 把数组的元素组成字符串，默认使用，连接

join(“符号”) 用指定的符号连接

3 reverse()

4 concat()

5 push() 把元素存在数组末尾

pop()取到数组的末尾元素，并移除数组中的这个元素

6 unshift() 把元素存在数组的开头

shift() 取到数组开头的元素，并移除数组中的这个元素

7 toString()

8 toLocaleString()

### 3.7 undefined(未定义类型)

访问一个对象的某个属性，而这个对象下没有这个属性，则会出现undefined

访问某个变量，这个变量不存在时，会出现undefined

## 4 一些函数

### 4.1 typeof关键字

作用：判断某个值或某个变量是什么类型。

var x = 10；

typeof x ---------> number

typeof [] ---------->object

typeof {} ---------->object

var x = function(){}

typeof x -----🡪function

那么如何判断是 数组还是对象？ 数组有length属性。判断是否有某属性，使用**hasOwnProperty**(‘length’)

### 4.2 for(key in ...)

var user = {id:1,name:'zhangsan',age:20};

for(key in user){

alert(key+"<<<<"+**user[key]**);

}

### 4.3 with关键字

作用：设置访问范围

|  |
| --- |
| var student ={  id:1,  name:'zhangsan',  age:20,  teacher:{  id:100,  name:'wangwu'  }  }  alert(student.id); 1  alert(student.teacher.id); 100  with(student.teacher){  alert(id); 100  } |

### 4.4 eval函数

作用：将字符串转换成js表达式。

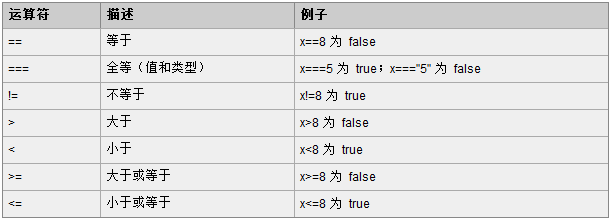
|  |
| --- |
| var str = "1+2";  alert(str);  var result = eval(str);  alert(result); ------3  var str ="{id:1,name:'zhangsan'}";  var user = eval(str);  alert(user); -------报错  var str ="{id:1,name:'zhangsan'}";  var user = eval("("+str+")");  alert(user); |

### 4.5 运算符

JavaScript运算符与Java运算符基本一致。

这里我们来寻找不同点进行学习

#### 4.5.1比较运算符



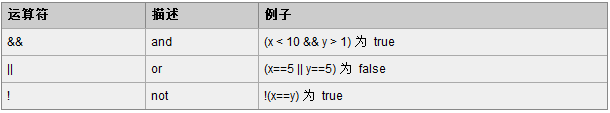
== 逻辑等。仅仅对比 数据值。11 “11”

**=== 全等**。 对比数据值并且对比类型。

如果值和类型都相同，则为true；值和类型有一个不同，则为false

|  |
| --- |
| 随堂练习：  var x=8;  alert(x==8); // true  alert(x===8); //true  alert(x=="8"); // true  alert(x==="8"); //false |

#### 4.5.2逻辑运算符



在JS里：没有单&和单|的用法

|  |
| --- |
| 课堂练习：  1、【多选题】在HTML文件中引入JavaScript有内嵌式和外联式两种方式，其中内嵌式，在HTML文档中，通过\_\_b\_\_\_\_\_标签引入，外联式通过 \_c\_\_\_\_\_ 标签引入.js文件  A. <JavaScript>  B. <script>  C. <script src=””>  D. < JavaScript src=””>  E. < JavaScript href=””>  F. <script href=””>  2、【单选题】JavaScript中以下代码报错的是（b）  A alert(123)  B alert(abc)  C alert(“abc”)  D 都报错  3、【单选题】JavaScript中如下代码，执行的结果是（b）  <script type="text/javascript">  var a;  alert(a);  </script>  A. null  B. undefined  C. 报错  D. 以上都不对 |

## 5 JavaScript中的变量

### 5.1全局变量

|  |
| --- |
| var x =100;  function fun1(){  x = x+100;  }  function fun2(){  alert(x);  }  fun1();  fun2(); //200 fun1 fun2 共享一个x |

### 5.2局部变量

|  |
| --- |
| function fun(){  var x = 100;  x = x+100;  }  alert(x);  function fun(){  alert(x); // undefined  if(true){  var x = 100;  }  alert(x); //100  } |

|  |
| --- |
| JS的局部变量，在定义它的方法中都可以使用（在定义后使用）  function fun(){  for( i=0;i<=100;i++){ }  alert(i); //101  }  fun(); |

## 6 JavaScript中的异常处理

关键字： try catch finally throw

|  |
| --- |
| function fun(num){  if(num<0){  throw {msg:'负数不能求阶乘！！！'};  }  if(num==1 || num==0){  return 1;  }else{  return num\*fun(num-1);  }  }  try{  var result = fun(-4);  alert(result);  }catch(e){  alert(e.msg);  } |

## 7 JavaScript中的Date

JS中的Date 和 Java中一样，都是记录的距离 1970-01-01 零时的毫秒数

var date = new Date(); //系统当前时间

var date = new Date(2000) //距离 1970-01-01 零时 2000毫秒

var date = new Date("1990-10-10"); 之前的IE 不支持 （IE8不支持）

var date = new Date("1990/10/10");

var date = new Date("1990/10/10 11:11:11");

var date = new Date("August 6 1990 11:11:11");

var date = new Date(1990,10,10,11,11,11);// 月份是 0---11

### 7.1 日期比较大小

var d1 = new Date("1990-10-10");

var d2 = new Date("1991/10/10");

alert(d2>d1); true

### 7.2 日期做运算

var d1 = new Date("1990/10/10");

var d2 = new Date("1991/10/10");

alert(d2-d1); 算出的是两个日期之间的毫秒数

### 7.3 Date的函数

setDate 设置 Date 对象中月份中的天数，其值介于 1 至 31 之间。

setHours 设置 Date 对象中的小时数，其值介于 0 至 23 之间。

setMinutes 设置 Date 对象中的分钟数，其值介于 0 至 59 之间。

setSeconds 设置 Date 对象中的秒数，其值介于 0 至 59 之间。

setTime 设置 Date 对象中的时间值。

setMonth 设置 Date 对象中的月份，其值介于 0 至 11 之间。

setYear

--------------------------------------------------------------------

getDate 返回 Date 对象中月份中的天数，其值介于 1 至 31 之间

getDay 返回 Date 对象中的星期几，其值介于 0 至 6 之间

getHours 返回 Date 对象中的小时数，其值介于 0 至 23 之间

getMinutes 返回 Date 对象中的分钟数，其值介于 0 至 59 之间

getSeconds 返回 Date 对象中的秒数，其值介于 0 至 59 之间

getMonth 返回 Date 对象中的月份，其值介于 0 至11 之间

getFullYear 返回 Date 对象中的年份，其值为四位数

getTime 返回自某一时刻（1970 年 1 月 1 日）以来的毫秒数

---------------------------------------------------------------------

toGMTString 使用格林尼治标准时间 (GMT) 数据格式将 Date 对象转换成字符串表示

toLocaleString 使用当地时间格式将 Date 对象转换成字符串表示

toLocaleDateString，按照本地格式显示一个日期对象的日期部分

toLocaleTimeString，按照本地格式显示一个日期对象的时间部分

--------------------------------------------------------------------------

Date.parse (date string )

用日期字符串表示自 1970 年 1 月 1 日以来的毫秒数

## 8 JavaScript中的正则表达式（Regular Expression regex）

正则表达式： 一个表达式，可以表示一个类的字符串。

作用： 1 判断某个字符串是否符合某种形式。

2 查找符合条件的子串

**JS中定义正则：** /正则表达式/

正则的规则：

|  |
| --- |
| . 表示任意一个字符  \d 表示一位数字  \D 非数字  \w 单词字符  \W 非单词字符 |

|  |
| --- |
| 1. z] 小写字母a到z 2. Z] 3. zA-Z]   [0-9]----------\d   1. zA-Z\_] \w 2. e]   [bc] |

次数：

|  |
| --- |
| ？ 0次或1次  \* 0次或多次  + 1次或多次  {m} 确定m次 \d{3} [a-z]{3}  {m,} 至少m次 \d{3,} [a-z]{3,}  {m,n} m到n次 \d{3,5} [a-z]{3,5} |

**例如：** /^[a-zA-Z]{5,10}$/

**作用：** 格式验证和提取子串

**1 格式验证：**在JS 如何判断某个字符串是否符合某个正则的格式(匹配正则)

var username = "abcde";

var usernameReg = /^[a-z]{5}$/;

两种方式：

方式一：

|  |
| --- |
| if(username.match(usernameReg)==null){  alert("用户名格式不正确")；  }else{  alert("格式正确");  } |

**注意：js中的格式验证时正则要使用 ^ 和 $定义**

方式二：

|  |
| --- |
| if(usernameReg.test(username)){  alert("格式正确")；  }else{  alert("格式不正确");  } |

**2 提取子串**

|  |
| --- |
| var reg = /aa\w/;  var reg2 = /aa\w/g  var reg3 = /aa\w/gi;  var str = "aaA aaa aab aac bbb abc AAA aAx";    var ss = str.match(reg); //得到第一个匹配这个正则的子串    var ss = str.match(reg2);// 得到所有匹配正则的子串,ss是数组    var ss = str.match(reg3);    **g------------全局匹配 globel**  **i------------忽略大小写 ignoreCase** |