# JavaScript概述

管理科学与工程学科
耿方方

# 主要内容

- 什么是JavaScript？
- JavaScript发展史
- JavaScript版本
- JavaScript作用
- JavaScript开发工具
- 在HTML中使用JavaScript
- JavaScript语法

# 什么是JavaScript？

概念

- JavaScript是一种为网站添加互动以及自定义行为的客户端脚本语言，因此通常只能通过Web浏览器去完成操作，而无法像普通意义上的程序那样独立运行。

# 什么是JavaScript？

主要特点

- 解释性执行的脚本语言

JavaScript的语法基本结构形式与C、C++、Java十分类似，但是在使用之前，不需要先编译，而是在程序执行中被逐行的解释。

- 简单弱类型脚本语言

JavaScript的简单性主要在于其基于Java基本语句和控制流之上的简单而紧凑的设计；其次在于其变量类型是采用弱类型，并未使用严格的数据类型。

# 什么是JavaScript？

主要特点

- 相对安全的脚本语言

JavaScript作为一种安全性语言，不被允许访问本地硬盘，且不能将数据存入服务器，不允许对网络文档进行修改和删除，只能通过浏览器实现信息浏览或动态交互，从而有效地防止数据的丢失或对系统的非法访问。

- 跨平台性的脚本语言

JavaScript依赖于浏览器本身，与操作环境无关，只要计算机能运行支持JavaScript的浏览器，就可正确执行，从而实现了跨平台的特性。

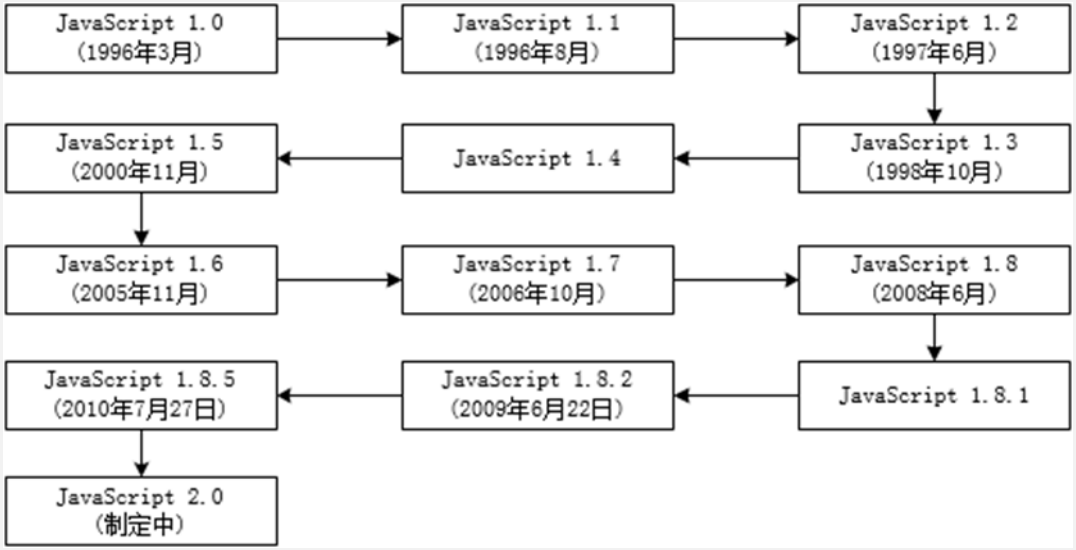
# JavaScript发展史

- JavaScript诞生于1996年。当时，它的目的是处理以前由服务器端语言（如Perl）负责的一些输入验证操作。当时，上网的速度仅为28.8kbit/s，在服务器端验证一次表单，再看到页面需要等待30s。Netscape公司决定开发一种客户端语言。
- 与Java没有任何的关系，它由Netscape公司与Sun公司合作开发。JavaScript最开始的名字是LiveScript，因当时Java风靡一时以及当时正与Sun公司进行合作等因素，于是将LiveScript改为了JavaScript。JavaScript的第一个版本，出现在1996年推出的NetScape Navigator 2浏览器中。

# JavaScript发展史

- 在微软进入市场后，有3种不同的JavaScript版本同时存在：Navigator中的JavaScript、IE中的Jscript及 CEnvi中的ScirptEase。
- 1997年，JavaScript1.1版本作为一个草案提交给欧洲计算机制造商协会（ECMA），最终由来自Netscape、Sun、微软、Borland和其他一些对脚本编程感兴趣的公司程序员组成了TC39委员会，该委员会被委派标准化一个通用、跨平台、中立于厂商的脚本语言的语法和语义。
- TC39委员会制定了“ECMA-262标准”，该标准由国际标准化组织采纳通过，作为各浏览器生产开发所使用的脚本程序的统一标准。

# JavaScript版本



# JavaScript作用

- 通常情况下，Web前端开发者使用JavaScript在给网页添加交互作用。网页的结构层是HTML；网页表现层由CSS构成；网页行为层由JavaScript组成。网页上的所有元素、属性和文本都能通过使用DOM（文本对象模型）的脚本来获得。
- Web前端开发者可通过JavaScript来实现改变网页内容、CSS样式、对用户输入做出反馈等操作。

# JavaScript开发工具



# 在HTML中使用JavaScript

- 用JavaScript编写的代码必须通过HTML/XHTML文档才能执行。目前有三种方法可以调用JavaScript。
- 方法一：将JavaScript代码放到文档<head>或<body>标签中的<script>标签之间。但最好的做法是将<script>标签放到HTML文档的最后，<body>结束标签之前。
- 方法二：将JavaScript代码存为一个扩展名为.js的独立文件。典型的做法是在文档的<head>部分放置一个<script>标签，并把它的src属性指向该文件。
- 方法三：直接将脚本嵌入到HTML标记的事件中

# 在HTML中使用JavaScript

- 方法一：<script>是HTML语言为引入脚本程序而定义的一个双标记。在网页中最常用的一种插入脚本的方法是使用<script></script>标记对。
- 具体方法是：把脚本标记对<script></script>置于网页的head部分或body部分中，然后在其中加入脚本程序。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>第一个 JavaScript 编程</title>
<script>
    //JavaScript 代码
</script>
</head>
<body>
<div id="bodyContent" class="body-content">
</div>
</body>
</html>
```

# 在HTML中使用JavaScript

- 案例1：
- <! DOCTYPE html>
- <head>
- <title>第一个JavaScript程序</title>
- </head>
- <body>
- <h1>第一个JavaScript程序</h1>
- <script>document.write( ‘<h3>Welcome!</h3>’ );
- </script>
- </body>
- </html>

# 在HTML中使用JavaScript

- 方法二：将JavaScript代码存为一个扩展名为.js的独立文件。典型的做法是在文档的<head>部分放置一个<script>标签，并把它的src属性指向该文件。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>第一个 JavaScript 编程</title>
<script src="file.js"></script>
</head>
<body>
<div id="bodyContent" class="body-content">
</div>
</body>
</html>
```

# 在HTML中使用JavaScript

- 案例2：
- <html>
- <head>
- <title>链接式</title>
- <meta charset="UTF-8">
- </head>
- <body>
- <h1>我的第一个JS程序</h1>
- <script src="demo2-01.js"></script>
- </body>
- </html>

Demo2-01.js

```
var today = new Date();
var hourNow = today.getHours();
var greeting;

if (hourNow > 18) {
    greeting = 'Good evening!';
} else if (hourNow > 12) {
    greeting = 'Good afternoon!';
} else if (hourNow > 0) {
    greeting = 'Good morning!';
} else {
    greeting = 'Welcome!';
}

document.write('<h3>' + greeting + '</h3>');
```

# 在HTML中使用JavaScript

- 方法三：可以直接在HTML某些标记内添加事件，然后将JavaScript脚本写在该事件的值内，以响应输入元素的事件。

- **案例3**：
- <html>
- <head>
- <title>第一个JavaScript程序</title>
- </head>
- <body>
- <p onclick=”javascript：alert（’Hello,the Web world！’);”>click here</p>
- </body>
- </html>

# 在HTML中使用JavaScript

- 浏览器在遇到<script>元素时会停止载入脚本，然后检查，看看自己是否有什么要做的。这样会影响页面的加载时间。
- 需要注意的是：上述代码中<script>标签没有包含传统的type="text/javascript"属性，是因为在HTML5规范中，script属性默认是text/javascript，所以可以省略，但是在HTML4.01和XHTML1.0规范中，type属性是必须的。

# JavaScript语法



- JavaScript程序使用Unicode字符集编写，它是一种区分大小写的语言，也就是说，在输入关键字、变量、函数名以及所有的标识符时，都必须采取一致的字母大小写格式。
- 标识符可以按照下列格式规则组合起来的一或多个字符：
  第一个字符必须是一个字母、下划线（_）或一个美元符号（$）；
  其他字符可以是字母、下划线、美元符号或数字。
  按照惯例，标识符采用驼峰大小写格式，也就是第一个字母小写，剩下的每个单词的首字母大写，例如：

```
firstSecond
myCar
doSomethingImportant
```

不能把关键字、保留字、true、false和null用作标识符。

# JavaScript语法



- JavaScript会忽略程序中记号之间的空格、制表符和换行符。因为可以在程序中任意使用空格、制表符和换行符，因此开发者可以采用整齐、一致的方式排版JavaScript代码，增加代码的可读性。
- JavaScript中的简单语句后面通常都有分号（;）主要是为了分割语句。

# JavaScript语法

## 注释

- JavaScript和Java一样，它也支持C++、C型的注释。JavaScript会把处于“//”和一行结尾之间的任何文本都当做注释忽略掉。此外“/*”和“*/”之间的文本也会被当做注释，这个注释可以跨越多行，但是其中不能有嵌套的注释。

```
//这是一条单行注释
/*
这是一个多行注释
我是第二行注释内容
我是第三行注释内容
*/
/*这是一条注释*/      //这是另一条注释
```

# JavaScript语法

## 关键字和保留字

- JavaScript具有特定用途的关键字，这些关键字可用于表示控制语句的开始或结束，或者用于执行特定操作。

表 14-01 保留的 JavaScript 关键字

| | | | | |
|---|---|---|---|---|
| break | do | If | switch | typeof |
| case | else | In | this | var |
| catch | false | instanceof | throw | void |
| continue | finally | New | true | while |
| default | for | Null | try | with |
| delete | function | return | | |

# JavaScript语法

## 关键字和保留字

- JavaScript中有可能将来被用作关键字的，也不能用作标识符的保留字。

表 14-02 ECMA 扩展保留的关键字

| | | | | |
|---|---|---|---|---|
| abstract | double | goto | native | static |
| boolean | enum | implements | package | super |
| byte | export | import | private | synchronized |
| char | extends | int | protected | throws |
| class | final | interface | public | transient |
| const | float | long | short | volatile |
| debugger | | | | |

- 此外还应该避免把JavaScript预定义的全局变量名或全局函数名用作标识符。避免使用的标识符如表所示。

表 14-03 要避免使用的其他标识符

| | | | | |
|---|---|---|---|---|
| arguments | encodeURI | Infinity | Object | String |
| Array | Error | isFinite | parseFloat | SyntaxError |
| Boolean | escape | isNaN | parseInt | TypeError |
| Date | eval | NaN | RangeError | undefined |
| decodeURI | EvalError | Number | ReferenceError | unescape |
| decodeURIComponent | Function | Math | RegExp | URIError |

# JavaScript语法

## 变量

- 变量是用于存储信息的容器：
- JavaScript的变量是一种弱类型变量，所谓弱类型变量是指它的变量无特定类型，定义任何变量都是用“var”关键字，并可以将其初始化为任何值，而且可以随意改变变量中所存储的数据类型。例如：

```
var sMyString
```

- 通过赋值语句向 JavaScript 变量赋值：

```
var x=5;
var carname="Volvo";
```

变量名在 = 符号的左边，而需要向变量赋的值在 = 的右侧。

在以上语句执行后，变量 x 中保存的值是 5，而 carname 的值是 Volvo。

- 可以使用一条语句定义多个变量，只要每个变量用逗号分隔开即可：

```
var message= “hi” ,
    found=false, age=29;
```

# JavaScript语法



- 实例4:

```
<html>
 <body>
 <script type="text/javascript">
     var firstname;
     firstname="George";
     document.write(firstname);
     document.write("<br />");
      firstname="John";
     document.write(firstname);
  </script>
  <p>上面的脚本声明了一个变量，为其赋值，显示该值，改变该值，然后再显示该值。</p>
</body>
</html>
```

# JavaScript语法

## 变量

- 在函数内声明了一个变量，就只能在该函数中访问该变量。当退出该函数后，这个变量会被撤销。这种变量称为局部变量。您可以在不同的函数中使用名称相同的局部变量，这是因为只有声明过变量的函数能够识别其中的每个变量。
- 如果您在函数之外声明了一个变量，则页面上的所有函数都可以访问该变量。这些变量的生存期从声明它们之后开始，在页面关闭时结束。这种变量成为全局变量。

例如:

```
    function test(){
          var message= “hi” ;    //局部变量
}
    test();
    alert(message);    错误!
```

# JavaScript语法



- JavaScript是一种弱类型语言，这意味着Web前端开发者可以在任何阶段改变变量的数据类型，而无需像强类型语言一样在声明变量的同时还必须同时声明变量的数据类型。
- JavaScript的三种基本数据类型有：字符串、数值、布尔值。除了这三种基本的类型外，还有数组、undefined、null、对象。

- 字符串由零个或多个字符构成。字符包括（但不局限于）字母、数字、标点符号和空格。字符串必须包在引号里面，单引号或双引号都可以。JavaScript可以随意的选用引号，但最好还是根据字符串所包含的字符来选择。即如果字符串包含双引号，就把整个字符串包含在引号里面；如果包含单引号就把整个字符放在双引号里面。

```
var mood="don't ask";
var mood='中国"飞人"勇夺金牌';
```

- 如果一个字符串中既有单引号又有双引号，那么这种情况下需要把那个单引号或双引号看做一个普通字符，而不是这个字符串的结束标志，这种情况下需要对这个字符进行转义，在JavaScript中用反斜线对字符串进行转义

```
var score=” run time 3\’ 15\” ”   //转义字符后面的第1个字符将按原样输出
```

- 下面的表格列出了其余的特殊字符，这些特殊字符都可以使用反斜杠来添加到文本字符串中：

| 代码 | 输出 |
| --- | --- |
| \r | 回车符 |
| \t | 制表符 |
| \b | 退格符 |
| \f | 换页符 |

| 代码 | 输出 |
| --- | --- |
| \’ | 单引号 |
| \″ | 双引号 |
| \& | 和号 |
| \\ | 反斜杠 |
| \n | 换行符 |

- 转换为字符串：第一种使用tostring()方法，数值、布尔值、对象和字符串值都有tostring()方法，但null和undefinded值没有这个方法。第二种是string()函数，这个函数可以将任何类型的值转换为字符串。

例如：

```
var age=11;
var ageAsString=age.toString();    //字符串“11”
var found=true;
var foundAsString=found.toString(); //字符串“true”

var value1=10,
    value2=true,
    value3=null, value4;
alert (string (value1) ) ;    // “10”
alert (string (value2) ) ;    // “true”
alert (string (value3) ) ;    // “null”
alert (string (value4) ) ;    // “undefinded”
```

- 案例5：

```
<html>
    <head>
        <title>字符串实例</title>
        <meta charset="UTF-8">
    </head>
    <body>
       <h1>字符串变量</h1>
      <div id="title">Howdy
      <span id="name">friend</span>!</div>
      <div id="note">Take a look around...</div>
    <script src="demo2-05-1.js"></script>
    </body>
</html>
```

Demo2-05-1.js

```
var username;
var message;
username = 'Molly';
message = 'See our upcoming
range';

var elName =
document.getElementById('name');
elName.textContent = username;

var elNote =
document.getElementById('note');
elNote.textContent = message;
```

- 如果想给一个变量赋一个数值，不用限定它必须是一个整数。JavaScript允许使用带有小数点的数值，并且允许任意位小数点，这样的数称为浮点数，数值主要数据类型如下所示。

```
var num=33.25                    //这是一个浮点数
num=-88;                         //这是一个负数
num=-20.333                      //这是一个负数浮点数
```

- 数值转换：把非数值转换为数值的三个函数：Number（）、parseInt（）和parseFloat（）
- Number（）可以用于任何数据类型。

```
var num1=Number（“Hello World!”）；   //NaN
var num2=Number（“ ”）；              //0
var num3=Number（“000011”）；         //11
var num4=Number（true）；             //1
```

- 可以使用parseInt（）和parseFloat（）进行转换，例如：

```
var  num1=parseInt（“AF”，16）       //175
var  num2=parseInt（“AF”     ）      //NAN
var  num3=parseFloat（“1234blue”）   //1234
var  num4=parseFloat（“22.5”）       //22.5
var  num5=parseFloat（“22.34.5”）    //22.34
```

- 案例6:

```
<html>
 <head>
     <title>数值变量</title>
     <meta charset="UTF-8">
 </head>
 <body>
     <h1>数值变量</h1>
 <div id="content">
   <h2>Custom Signage</h2>
   <div id="cost">Cost: $5 per tile</div>
 </div>
 <script src="demo2-06-1.js"></script>
</body>
 </body>
</html>
```

Demo2-06-1.js

```
var price;
var quantity;
var total;

price = 5;
quantity = 14;
total = price * quantity;

var el = document.getElementById('cost');
el.textContent = '$' + total;
```

- 布尔数据只有两个可选值：true或者false。
- 布尔值不是字符串，不能将布尔值用引号括起来。布尔值的false与字符串值“false”是完全不相关的两码事。

```
var married=false;                    //变量 married 设置为布尔值 false
var married="false";                  //变量 married 设置为字符串“false”
```

# 数据类型-布尔值

- 所有类型的数值都有与Boolean值等价的值， 要将一个值转换为其对应的Boolean值，可以调用转型函数Boolean（）。对应关系如表

| 数据类型 | 转换为true的值 | 转换为false的值 |
| --- | --- | --- |
| Boolean | true | false |
| String | 任何非空字符串 | “”（空字符串） |
| Number | 任何非零数字值（包括无穷大） | 0和NAN |
| Object | 任何对象 | null |
| Undefined | n/a（不适用） | undefined |

- 案例7

```
<html>
    <head>
        <title>布尔变量</title>
        <meta charset="UTF-8">
    </head>
    <body>
        <h1>布尔变量</h1>
        <script>
    var married=true;
    document.write(typeof(married)+"<br/>");   // 输出boolean
    married="true";
    document.write(typeof(married));        // 输出string
</script>

    </body>
</html>
```

# 数据类型-Undefined

- Undefined类型只有一个值，即特殊的Undefined。在使用var声明变量但未对其加以初始化时，这个变量的值就是undefined。

- ```
  var message;
  alert(message==undefined);  //true
  ```
  ```

# 数据类型-null

- null类型只有一个值，即特殊的null。从逻辑角度讲，null值表示一个空对象指针。

- ```
  var car=null;
  alert(typeof car);  //object
  ```
  ```

- 字符串、数值和布尔型数据都属于离散值，即在任意时刻只能存储一个值。如果想用一个变量来存储一组值，就需要使用数组。
- 数组是由名称相同的多个值构成的一个集合，集合中的每个值都是这个数组的元素。例如可以使用数组变量rank来存储论坛用户所有可能的级别。

创建数组的方法

方法一：

同创建任何其他变量的方法类似，可以直接创建数组并命名，赋给数组的值被包含在一对中括号里面，每个值用逗号隔开。可以在同一数组中存储字符串、数字和布尔值。这种方法为字面量法。

例如：

```
var colors;
colors=[ ‘white’ ,  ‘black’ ,   ‘custom’ ]
var el=document.getElemnetById( ‘color’ );
el.textContent=colors[0];
```

方法二：
数组构造函数：这种方法的形式是用一个new关键字，后面跟着Array（）；值在圆括号中指定，值之间用逗号分隔。还可以调用叫做item（）的方法来从数组中获取数据。
例如：

```
var colors=new Array（‘white’，‘black’，‘custom’）；
var el=document.getElemnetById（‘color’）;
el.innerHtml=colors[0];
```

数组的length属性：该属性用于返回数组的长度。
语法：
array.length
例如：创建一个数组对象，并获取该数组对象的长度，代码如下：

```
var arr=new Array (1, 2, 3, 4, 5) ;
document.write(arr.length);
```

运行结果：5
说明：当使用new Array ()创建数组时，在不对其进行赋值的情况下，length属性的返回值是0。
Array对象的下标是从0开始的。

数组的方法：
1、concat（）方法：
该方法用于将其他数组连接到当前数组的尾端。
语法：arrayObject.concat(arrayX,arrayX,•••,arrayX)
arrayObject:必选项。数组名称。
arrayX：必选项。该参数可以是具体的数组元素值，也可以是数组对象。
例如：

```
Var arr=new Array (1, 2, 3, 4) ;
Document.write(arr.concat(6,7,8,9));
```

数组的方法：
2、shift（）方法：
该方法用于把数组中的第一个元素从数组中删除，并返回删除元素的值。
语法：arrayObject.shift（）
arrayObject:必选项。数组名称。
例如：

```
Var arr=new Array (1, 2, 3, 4) ;
Var del=arr.shift();
Document.write( ‘删除元素为：’ +del+ ‘; 删除后的数组为：’ +arr);
```

3、pop（）方法：
该方法用于把数组中的最后一个元素从数组中删除，并返回删除元素的值。
语法：arrayObject.pop（）
arrayObject:必选项。数组名称。

数组的方法：

4、push（）方法：

该方法向数组的末尾添加一个或多个元素，并返回添加后的数组长度。

语法：

arrayObject.push(newelement1,newelement2,newelement3…)

5、unshift（）方法：

该方法向数组的开头添加一个或多个元素，并返回添加后的数组。

例如：

```
var arr=new Array (4, 5, 6) ;
document.write( ‘原数组’ +arr+ ‘<br/>’ );
arr.unshift(1,2,3);
Document.write( ‘新数组’ +arr);
```

数组的排序:
1、reverse () 方法
该方法用于颠倒数组中的元素顺序。
语法:
arrayObject.reverse();
2、sort () 方法
该方法用于对数组的元素进行排序。
语法:
arrayObject.sort();
获取数组中的某段数组元素
语法:
arrayObject.slice (start,end)

案例8：

```
<html>
    <head>
        <title>数组</title>
        <meta charset="UTF-8">
</head>
    <body>
        <h1>数组</h1>
    <div id="content">
      <div class="message">
Color: <span id="colors"></span></div>
    </div>
    <script src="demo2-08-1.js"></script>
    </body>
</html>
```

```
var colors = ['white', 'black', 'custom'];
colors[2] = 'beige';

var el =
document.getElementById('colors');
el.textContent = colors[2];
```

# JavaScript语法



- 表达式是JavaScript的一个“短语”，JavaScript解释器可以计算表达式，从而生成一个值，最简单的表达式是直接量或变量名。
- 表达式可以求出一个值，求值的过程可以包含运算。基本上有两种类型的表达式。
- 一专门给变量赋值的表达式

```
var color= ‘beige’ ;
```

- 二表达式中可以使用两个或更多个值，表达式最终会返回一个值。

```
var area=3*2;
```

# JavaScript语法



- 算术运算符：JavaScript中，加减乘除都是一种操作，这些算术操作中的每一种都必须借助于相应的操作符才能完成，操作符是JavaScript为完成各种操作而定义的一些符号。假设y=3，则：

表 14-04 JavaScript 算数运算符

| 运算符 | 描述 | 例子 | 结果 |
|---|---|---|---|
| + | 加 | x=y+2 | x=5 |
| - | 减 | x=y-2 | x=1 |
| * | 乘 | x=y*2 | x=6 |
| / | 除 | x=y/2 | x=1.5 |
| % | 求余数 | x=y%2 | x=1 |
| ++ | 累加 | x=++y | x=4 |
| -- | 递减 | x=--y | x=2 |

# JavaScript语法



- 赋值运算符用于给JavaScript变量赋值，假设x=6，y=3，则：

表 14-05 JavaScript 赋值运算符

| 运算符 | 例子 | 等价于 | 结果 |
|---|---|---|---|
| = | x=y | | x=3 |
| += | x+=y | x=x+y | x=9 |
| -= | x-=y | x=x-y | x=3 |
| *= | x*=y | x=x*y | x=18 |
| /= | x/=y | x=x/y | x=2 |
| %= | x%=y | x=x%y | x=0 |

- 加号（+）是一个比较特殊的操作符，它既可以用于数字，也可用于字符串，代码如下所示。

```
var message="I am feel "+"good";
```

- 比较运算符：给定 x=5，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;<br>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 |

- 逻辑操作符：整合多个表达式，返回true或false

| 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 |

# JavaScript语法

## 运算符

案例9:

```
<html>
    <head>
        <title>算术运算符</title>
        <meta charset="UTF-8">
</head>
    <body>
       <h1>算术运算符</h1>
  <div id="content">
      <div class="message number">Subtotal: $<span id="subtotal"> </span></div>
      <div class="message number">Shipping: $<span id="shipping"> </span></div>
      <div class="message number">Total: $<span id="total"> </span></div>
  </div>
  <script src="demo2-09-1.js"></script>
    </body>
</html>
```

```
var subtotal = (13 + 1) * 5;
var shipping = 0.5 * (13 + 1);

var total = subtotal + shipping;
var elSub =
document.getElementById('subtotal');
elSub.textContent = subtotal;

var elShip =
document.getElementById('shipping');
elShip.textContent = shipping;

var elTotal =
document.getElementById('total');
elTotal.textContent = total;
```

# 想一想？

- 1、使用字符串运算符“+”，连接字符串，并使用document.write语句输出个人信息。
- 2、已知长方形的长和宽，根据长方形面积公式，应用算术运算符计算出长方形的面积。