第 1 章 JavaScript 语言入门

1 为什么学习 JavaScript

提要：Javascript 是学习脚本语言的首选。她兼容性好，绝大多数浏览器均支持 Javascript，而且她功能强大，实现简单方便，入门简单，即使是程序设计新手也可以非常快速容易地使用 JavaScript 进行简单的编程。

Javascript 是由 Netscape 公司创造的一种脚本语言。为便于推广，被定为 javascript，但是 javascript 与 java 是两门不相干的语言，作用也不一样。作为一门独立的编程语言，javascript 可以做很多的事情，但它最主流的应用还是在 Web 上——创建动态网页(即网页特效)。Javascript 在网络上应用广泛，几乎所有的动态网页里都能找到它的身影。目前流行的 AJAX 也是依赖于 Javascript 而存在的。

Javascript 与 Jscript 也不是一门相同的语言，Jscript 和 vbscript 是微软开发的两种脚本语言，微软，Netscape 公司以及其他语言开发商为减少 web 开发者的兼容麻烦，所以成立 ECMA，该组织专门制定脚本语言的标准和规范。ECMA 制定的标准脚本语言叫做 ECMAScript，Javascript 符合 ECMA 的标准，其实 Javascript 也可以叫做 ECMAScript.

Jscript 也 ECMA 的标准，但用户较少。vbscript 仅局限在微软的用户，Netscape 不支持。

概括地说，JavaScript 就是一种基于对象和事件驱动，并具有安全性能的脚本语言，脚本语言简单理解就是在客户端的浏览器就可以互动响应处理程序的语言，而不需要服务器的处理和响应，当然 JavaScript 也可以做到与服务器的交互响应，而且功能也很强大。而相对的服务器语言像 asp asp.net php jsp 等需要将命令上传服务器，由服务器处理后回传处理结果。对象和事件是 JavaScript 的两个核心。

JavaScript 可以被嵌入到 HTML 文件中，不需要经过 Web 服务器就可以对用户操作作出响应，使网页更好地与用户交互；在利用客户端个人电脑性能资源的同时，适当减小服务器端的压力，并减少用户等待时间。

2 将 JavaScript 插入网页的方法

与在网页中插入 CSS 的方式相似，使用<script>标签在网页中插入 Javascript 代码。

使用下面的代码可以在网页中插入 JavaScript：

```
<script type="text/JavaScript" language="javascript">
    ...
</script>
```

language="javascript"表示使用 JavaScript 脚本语言，脚本语言还有 vbscript、jsscript 等，如果没有 language 属性，表示默认使用 JavaScript 脚本。其中的...就是代码的内容。例如：

```
<script type="text/JavaScript">
  document.write("我是菜鸟我怕谁!");
```

```
</script>
```

JavaScript 使用 document.write 来输出内容。将会输出在网页上输出：我是菜鸟我怕谁!

学过编程的人应该知道，“我是菜鸟我怕谁!”两侧双引号代表字符串的意思。不过不理解这个概念也无所谓，学到后面就知道什么时候应该加双引号，什么时候不需要加了。

有些浏览器可能不支持 JavaScript，我们可以使用如下的方法对它们隐藏 JavaScript 代码。

```
<html>
  <body>
  <script type="text/JavaScript">
  <!--
  document.write("我是菜鸟我怕谁!");
  //-->
  </script>
  </body>
</html>
```

<!-- -->里的内容对于不支持 JavaScript 的浏览器来说就等同于一段注释，而对于支持 JavaScript 的浏览器，这段代码仍然会执行。至于“//”符号则是 JavaScript 里的注释符号，在这里添加它是为了防止 JavaScript 试图执行-->。不过通常情况下，现在的浏览器几乎都支持 JavaScript，即使是不支持的，也会了解如何合理地处理含有 JavaScript 的网页。

JavaScript 的插入位置不同，效果也会有所不同，还可以像 CSS 一样，将 JavaScript 保存成一个外部文件，这些内容会在下一节讨论。

3 插入 JavaScript 的位置

JavaScript 脚本可以放在网页的 head 里或者 body 部分，而且效果也不相同。

（1）放在 body 部分的 JavaScript 脚本在网页读取到该语句的时候就会执行，例如：

```
<html>
<body>
<script type="text/JavaScript">
<!--
document.write("我是菜鸟我怕谁!");
//-->
```

```
</script>
</body>
```

（2）在 head 部分的脚本在被调用的时候才会执行，例如：

```
<html>
<head>
<script type="text/JavaScript">
....
</script>
</head>
```

通常是在<script>...</script>定义函数，通过调用函数来执行 head 里的脚本。

（3）也可以像添加外部 CSS 一样添加外部 JavaScript 脚本文件，其后缀通常为.js。例如：

```
<html>
<head>
<script src="scripts.js"></script>
</head>
<body>
</body>
</html>
```

如果很多网页都需要包含一段相同的代码，那么将这些代码写入一个外部 JavaScript 文件是最好的方法。此后，任何一个需要该功能的网页，只需要引入这个 js 文件就可以了。

注意：脚本文件里头不能再含有<script>标签。

注：放在 body 里的函数是一个例外，它并不会被执行，而是等被调用时才会执行。关于函数与调用的概念将在后面讲到。

4 JavaScript 语句

本节介绍 Javasctipt 中的语句，语句是编程的一个基本概念。

先来看看我们之前例子。

```
<script type="text/JavaScript">
<!--
document.write("我是菜鸟我怕谁!");
//-->
```

```
</script>
```

例子中的 document.write("我是菜鸟我怕谁!");

就是一个 JavaScript 语句，它可以告诉浏览器做出一个特定任务。特别的，这个语句是叫浏览器书写出一段内容。我们已经知道，document.write 的功能是输出文本，所以上面那个语句就是让浏览器输出“我是菜鸟我怕谁!”。

在 JavaScript 中，一行的结束就被认定为语句的结束。但是最好还是要在结尾加上一个分号“;”来表示语句的结束。这是一个编程的好习惯，事实上在很多语言中句末的分号都是必须的。

看看下面这个代码块

```
<script type="text/javascript">
{
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
}
</script>
```

5 JavaScript 注释

HTML、CSS 里都有注释，JavaScript 里自然也有注释，而且分为单行注释与普通注释两种。

5.1 单行注释

插入单行注释的符号是“//”

```
<script type="text/javascript">
// 我是注释，我是注释
document.write("我是菜鸟我怕谁？");
</script>
```

5.2 多行注释

多行注释以"/*"开始，以"*/"结束。

```
<script type="text/javascript">
```

```
/*
谁说菜鸟不会编程？
菜鸟不但会编程
还有书写注释的良好习惯
*/
document.write("我是菜鸟我怕谁？");
</script>
```

相信你已经了解注释的作用了。HTML 的注释不会被浏览器作为 HTML 解释，CSS 注释也不会被浏览器解释。同理，JavaScript 的注释也不会被执行。

注释的作用就是记录自己在编程时候的思路，以便以后自己阅读代码时可以马上找到思路。同样，注释也有助于别人阅读自己书写的 JavaScript 代码。总之书写注释是一个良好的编程习惯。

6 JavaScript 变量

（1）什么是变量

在代数中，我们会遇到下面的基础问题，如果 a 的值为 5，b 的值为 6，那么 a 与 b 的和是多少？在这个问题中，我们就可以把 a 和 b 看做变量，再设置一个变量 c 来保存 a 与 b 的和。

那么，上面的这个问题就可以用如下的 JavaScript 代码表示：

```
<script type="text/javascript">
// 计算 a + b 的和
a = 5;//给变量 a 赋值
b = 5;//给变量 b 赋值
c = a + b;//c 为 a + b 的和
document.write(c);//输出 c 的值
</script>
```

执行结果：10

在上面的例子中，我们用到了三个变量：a，b，c。这些都是变量的名字，在 JavaScript 中，我们需要用变量名来访问这个变量。在 JavaScript 中，变量名有如下规定：

- 变量名区分大小写，A 与 a 是两个不同变量。
- 变量名必须以字母或者下划线开头。

（2）声明变量

我们可以用 var 声明变量,比如：

```
<script type="text/javascript">
var a ; //声明一个变量 a
a = 5 ; //给变量赋值
</script>
```

其实在第一个例子中我们已经看到了，JavaScript 中可以不声明变量直接赋值。不过先声明变量是一个良好的编程习惯。

（3）给变量赋值

来看一个上面出现过的语句：

```
a = 5 ;
```

这个语句怎么读出来呢？“a 等于 5”？

对不起，不对。应该是“让 a 等于 5”，说得专业一点，这叫“给 a 赋值”。

我们之前已经提到过，a 是变量，是可以变的，所以从某种角度来说，它不等于任何值，只是暂时的等于某个值。来看下面这个例子，进一步熟悉一下赋值与等于的关系：

```
a = 5;//让 a 等于 5，a 的值暂时等于 5
a = 6;//让 a 等于 6，a 的值暂时等于 6
```

再次强调，如果 JavaScript 是你学习的第一门编程语言，一定要注意区分“等于”和“赋值”这两个不同的概念。

（4）变量的数据类型

其实，在 JavaScript 中，变量是无所不能的容器，你可以吧任何东西存储在变量里，例如：

```
var quanNeng1 = 123;//数字
var quanNeng2 = "一二三"//字符串
```

其中，quanNeng2 这个变量存储了一个字符串，字符串需要用一对引号括起来。变量还可以存储更多的东西，例如数组，对象，布尔值等等，我们会在后面介绍这些内容。

7 JavaScript 操作符

7.1 运算操作符

操作符是用于在 JavaScript 中指定一定动作的符号，其中算术操作符主要用来完成类似加减乘除的工作。

（1）操作符举例

看下面这段 JavaScript 代码。

```
c = a + b;
```

其中的"="和"+"都是操作符。

JavaScript 中还有很多这样的操作符，例如，加减乘除是 JavaScript 中比较基本的几个操作符，它们的意义与在数学中没有什么差别。

JavaScript 中最常见的操作符是赋值操作符“=”，上一节我们已经强调过，它不是等于。

（2）操作符的优先级

我们都知道，在数学中，“a + b * c”这个式子中，乘法将先于加法运算。同样，在 JavaScript 中，这个式子会按相同的顺序执行。我们称之为“优先级”，即“*”的优先级高于“+”。

与数学中一样，改变运算顺序的方法是添加括号，JavaScript 中改变优先级的方法也是添加括号。例如：

```
(a +b) * c
```

（3）字符串的连接

在 JavaScript 中，“+”不知代表加法，同样也可以使用它来谅解两个字符串，例如：

```
example = "乌" + "龟";
```

在上面的例子中，example 将包含“乌龟”这个字符串。这是由于“+”完成了“乌”和“龟”的连接，当然了，你也可以把这种行为理解成字符串的加法。

（4）自加一，自减一操作符

这里我们来看两个非常常用的运算符，自加一“++”；自减一“--”。首先来看一个例子：

```
a = 5;
a++;//a 的值变为 6
a--//a 的值有变回 5
```

上面的例子中，a++使得 a 的值在原来的基础上增加 1，a--则让 a 在现在的基础上在减去 1。所以，其实“a++”也可以写成

```
“a = a + 1”;//等同于 a++
```

（5）复合操作符

延续上面的例子，其实“a = a + 1”还可以写成：

```
a +=  1;//将 a 的值加 1 之后再赋给 a
```

这样把运算和赋值结合到一起的操作符叫做复合操作符。上面我们看到的是加法与赋值的结合，JavaScript 中还有其它的符合运算符：

```
a += b;// a = a + b
a -= b;// a = a - b
a *= b;// a = a * b
a /= b;// a = a / b
```

7.2 比较操作符和逻辑操作符

操作符是用于在 JavaScript 中指定一定动作的符号。

（1）比较操作符

比如：“==”符号，它是一个比较操作符，它表示的意思就是“相等吗？”。

例如：a==b 表示：“a 与 b 的值相等吗？”

在 JavaScript 中，这样的比较操作符有很多，下面就列出这些操作符以及它们的含义。

- “>” —— a 大于 b 吗？
- “<” —— a 小于 b 吗？
- “>=” —— a 大于等于 b 吗？
- “<=” —— a 小于等于 b 吗？
- “==” —— a 等于 b 吗？
- “!=” —— a 不等于 b 吗？

（2）逻辑操作符

数学里面的“a>b”在 JavaScript 中还表示为 a>b；数学中的“b 大于 a，b 小于 c”是“a<b<c”，那么在 JavaScript 中是不是也一样呢？对不起，JavaScript 没有那么聪明，你需要这么写：

```
b>a && a < b
```

你可能已经猜到了，“&&”是而且的意思。

```
if(条件 1 && 条件 2)
{//代码}
```

只有条件 1、2 同时满足，代码才会得到执行。

类似的操作符还有“或者(||)”和“非(!)”

```
if(条件 1 || 条件 2)
{//代码}
```

“||”表示或者的意思，只要条件 1 或 2 中有一个满足，代码就会得到执行。

```
if(!条件)
{//代码}
```

“！”表示非，也就是不是的意思，只有条件不满足的时候，代码才会得到执行。

8 JavaScript 流程控制语句

8.1 if else 语句（如果，否则）

if else 是所有编程语言里都有的功能，它使得程序具有简单的判断能力。

在介绍 if 之前，让我们先来了解一下布尔值这个概念。

(1)布尔(Bool)值

在变量一节中，我们曾经说过，变量可以用来存储布尔值。哪么布尔值的作用究竟是什么呢？简单的说，布尔值的作用就是用来表示“真的假的”。所以布尔值其实只有两种取值：真(true)和假(false)。

(2) if else 结构

其实“if else”的意思和字面意思是一样的，就是“如果”、“否则”。还是让我们来看一个使用 if 的例子吧。

```
<script type="text/JavaScript">
Var hobby = “VbScript”;
if (hobby == “JavaScript”)
{
document.write("有发展");
}
</script>
```

我们来解释一下这段代码。首先是一个“if”，它后面紧跟着一个括号，括号里则是一个条件，确切地说是一个布尔值。当条件成立的时候，这个值是 true，“{}”里的语句将会得到执行；否则这个值是 flase，“{}”里的语句将被忽略。

具体到我们的例子，因为 hobby 变量的值是“VbScript”，所以不做回答。如果 hobby 变量的值是“JavaScript”，则回答“有发展”。注意“==”这个符号，这个符号用来判断左右两边是否相等。下一节会详细地介绍。

如果你的爱好不是 JavaScript，那么没有任何输出。如果你希望它能对这种情况做出反应，我们可以请 else 来帮忙，看下面的代码：

```
<script type="text/JavaScript">
var hobby = "JavaScript"
if (hobby == "JavaScript")
{
document.write("有发展");
}
else//如果爱好不是 JavaScript
{
document.write("没有评价……");
}
</script>
```

上面的代码用到了“else”，它会给 if 添加一种“否则”的状态。当 hobby 不是“JavaScript”的时候，它会表明“没有评价”。

（3）if else 嵌套

如果想做更多的判断，可以用 if 的嵌套,看下面的代码。

```
<script type="text/JavaScript">
var hobby = "JavaScript"
if ( hobby == "JavaScript")
{
document.write("有发展");
}
else if ( hobby == "football")//如果爱好是足球。
//注意：这个 if 是嵌套在上一个 if else 中的 else 中的
{
document.write("我 X");
}
else//既不是 JavaScript 又不是足球
{
document.write("没有评价……");
}
</script>
```

第二个 if 只有在第一个 if 的条件不成立的时候才有机会执行。

8.2 JavaScript Switch 语句

当有很多种选项的时候，switch 比 if else 使用更方便。

上一节我们已经看到，利用 if else 可以让程序具有基本的判断能力，而使用嵌套的 if else 则可以让程序对多种情况进行判断。但是当情况的种类比较多的时候，使用 switch 语句将更加合适。

比如要实现如下功能的程序：输入一个学生的考试成绩，我们按照每十分一个等级将成绩分等，程序将根据成绩的等级做出不同的评价。

很明显，用 if else 可以实现这样的程序，但是代码会很复杂。而如果使用 switch 语句，代码则会简单一些，首先来看一下思路，再把它翻译成 JavaScript。

思路：

1. 将分数转化为特定等级以便于 switch 处理。
2. 判断分数属于哪种等级。
3. 根据分数等级做出评价：例如低于 60 给出挂科评价。

翻译成 JavaScript 就是如下代码(注意注释)：

```
<script type="text/JavaScript">
//首先，我们用 score 变量来存储分数，假设为 65
 var score = 65;
```

```
//用分数除以 10，parseInt 的作用是把它转换为整数，
//暂时不用深究，()内最后的结果为 6
 switch (parseInt(score / 10)) {
//switch 开始实现判断过程，case 6 得到满足
 case 0:
 case 1:
 case 2:
 case 3:
 case 4:
 case 5:
//根据不同的等级做出不同的行为。
//冒号后面的语句就是行为
//case0 到 5 的行为都是下面这个语句
 degree = "恭喜你，又挂了！";
 break;
 case 6:
 degree = "勉强及格";
 break;
 case 7:
 degree = "凑合，凑合"
 break;
 case 8:
 degree = "不错，不错";
 break;
 case 9:
 case 10:
 degree = "高手高手，佩服佩服";
 }//end of switch
 </script>
```

记得在每个 case 所执行的语句里添加上一个 break 语句。为了理解 break 的作用，我们来看看如果没有 break 会怎么样：

```
<script type="text/JavaScript">
 switch (parseInt(score / 10)) {
 case 0:
 case 1:
 case 2:
 case 3:
 case 4:
 case 5:
 degree = "恭喜你，又挂了！";
```

```
case 6:
degree = "勉强及格";

case 7:
degree = "凑合，凑合"
case 8:
degree = "8 错，8 错";
case 9:
case 10:
degree = "高手高手，佩服佩服";
}//end of switch
</script>
```

在上面的代码中，如果成绩是 50 分，那么 score/10 就是 5，则 case 5 后面的语句将会得到执行，同样，case6、7 等等后面的语句都会得到执行。也就是说，我们会得到：”恭喜你，又挂了！勉强及格凑合，凑合 8 错，8 错高手高手，佩服佩服“这样没有意义的评价。

这就是 swtitch 语句的执行逻辑，当发现某个 case 满足后，该 switch 中在该 case 后的所有语句都会得到执行。第一个例子中的 break 就是为了让 switch”停下来“。

8.3 JavaScript For 循环

所谓循环，就是重复执行一段代码。

前面我们已经看到了，if else 和 switch 是 JavaScript 具有了判断的能力，但是电脑的判断能力和人比起来差远了。电脑更擅长一件事情——不停地重复。我们在 JavaScript 中把这叫做循环。

for 语句结构如下：

```
for(初始条件;判断条件;循环后动作)
{
    循环代码
}
```

让我们来看一个简单的例子吧：有十个菜鸟报数，“菜鸟 1 号、菜鸟 2 号”。有了 for 循环，很少的代码就可以实现十个菜鸟的报数。

```
<script type="text/JavaScript">
var i=1;
```

```
for (i=1;i<=10;i++)
{
document.write("菜鸟"+i+"号<br />");
}
</script>
```

结果如下：

```
菜鸟 1 号
菜鸟 2 号
菜鸟 3 号
菜鸟 4 号
菜鸟 5 号
菜鸟 6 号
菜鸟 7 号
菜鸟 8 号
菜鸟 9 号
菜鸟 10 号
```

在上面那个例子中，循环恰好执行了 10 次，那么和“for (i=1;i<=10;i++)”一句中的 10 是不是 10 次的意思呢？下面我们就来看看 for 循环的工作机制。

首先"i=1"叫做初始条件，也就是说从哪里开始，特别的，我们的例子从 i=1 开始。

出现在第一个分号后面的"i<=10"表示判断条件，每次循环都会先判断这个条件是否满足，如果满足则继续循环，否则停止循环，继续执行 for 循环后面的代码。你可能想问了，我们设定了 i=0，岂不是永远都小于等于 10 吗？来看第三个部分。

最后的"i++"表示让 i 在自身的基础上加 1，这时每次循环后的动作.也就是说，每次循环结束，i 都会比原来大 1，执行若干次循环之后，i<=10 的条件就不满足了，这时循环结束。for 循环后面的代码将得到执行。

8.4 JavaScript While 循环

while 循环重复执行一段代码，直到某个条件不再满足。

（1）while 循环的结构：

```
while(判断条件)
{
        循环代码
```

```
 }
```

其实 while 循环和 for 循环的作用都是重复执行代码，例如下面这段代码，和上一节 for 循环的输出结果完全没有区别。先来读读例子的代码，下面会解释为什么这个循环会和上一节的 for 循环等价。

```
<html>
<body>
<script type="text/JavaScript">
var i=0;
while (i<=10)
{
document.write("菜鸟"+i+"号");
document.write("<br />");
i=i+1;
}
</script>
</body>
</html>
```

看起来好像比 for 循环少了点东西啊，只有一个判断条件啊。其实这个循环也是有初始条件的，只不过在之前就已经定义好了，例如上面例子中的"var i=0;"，至于变量 i 的增大，则是放到了循环体里面，其实这个过程和 for 没有什么区别，也是变量 i 不断变大，直到判断条件不满足，则循环结束。

（2）do while 循环的结构

do wile 结构的基本原理和 while 结构是基本相同的，但是它保证循环体至少被执行一次。因为它是先执行代码，后判断条件。例子如下：

```
<script type="text/JavaScript">
     i = 0;
     do
     {
     document.write("The number is " +  i);
     document.write("<br />");
     i++;
     }
     while (i <= 5)
</script>
```

8.5 JavaScript Break 与 Continue

Break 可以跳出循环，Continue 跳过本次循环。

（1）break

break 语句可以让循环中途停止，直接执行后面的代码。格式如下：

```
while (i<10)
{
if(特殊情况)
break;
循环代码
}
```

那么，当特殊情况发生的时候，循环就会立即结束。看看下面的例子，我们的菜鸟 7 号到 10 号在寝室打游戏……

```
<html>
<body>
<script type="text/JavaScript">
var i=0;
for (i=0;i<=10;i++)
{
if (i==6)
{
break;//如果 i 是 6 的话就退出循环。
}
document.write("菜鸟"+i+"号<br />");
}
</script>
</body>
</html>
```

当 i=7 的时候循环就会结束，不会输出后面循环的内容。

（2）continue

continue 的作用是仅仅跳过本次循环，而整个循环体继续执行。它的格式如下：

```
while (判断条件)
{
```

```
if(特殊情况)
continue;
循环代码
}
```

上面的循环中，当特殊情况发生的时候，本次循环将被跳过，而后续的循环则不会受到影响，来看看下面的例子：菜鸟 6 号外出学习 JavaScript 去了。

```
<html>
<body>
<script type="text/JavaScript">
var i=0
for (i=0;i<=10;i++)
{
if (i==3)
{
continue;
}
document.write("The number is " + i);
document.write("<br />");
}
</script>
</body>
</html>
```

上面的代码中，i=6 的那次循环将被跳过.

8.6 JavaScript For...In 循环

Javascript 中的 for in 循环通常用来遍历数组

首先要了解什么是数组，所谓数组，其实就是一个保存了一组类似变量的一个集合。我们来看一个保存了爱好的数组实例：

```
<html>
<body>
<script type="text/JavaScript">
var x;
var hobbies = new Array();//创建一个新的数组
hobbies[0] = "JavaScript";
hobbies[1] = "CSS";
```

```
hobbies[2] = "篮球";
for (x in hobbies)//数组中的每一个变量
{
document.write(hobbies[x] + "<br />");
}
</script>
</body>
</html>
```

输出结果如下：

```
JavaScript
CSS
篮球
```

我们来分析一下上面的例子：

var hobbies = new Array();一句创建了一个新的数组。

hobbies[0] = "JavaScript";以及之后的两句则是给 hobbies 数组赋值。这与我们之前见过的变量赋值不太一样，hobbies 后面多出一个"[0]"，这个是变量的索引。我们之前已经说了，数组是变量的集合，因此我们在赋值之前需要指明给数组中的哪一个变量赋值。在这里，"[0]"表示的是 hobbies 数组所包含的第一个变量，没错，数组的索引是从 0 开始的，开始可能有点别扭，慢慢就习惯了。

最后的 for in 循环就很好理解了，

```
for (x in hobbies)//数组中的每一个变量
     {
     document.write(hobbies[x] + "<br />");
     }
```

表示遍历 hobbies 数组的所有变量，并且将他们逐一输出。

9 JavaScript 弹出框

JavaScript 中有三种弹出框:警告(alert)、确认(confirm)以及提问(prompt)。

9.1 警告(alert)

在访问网站的时候，你遇到过这样的情况吗？“咚”的一声，一个小窗口出现在你面前，上面写着一段警示性的文字，或是其它的提示信息。如果你不点击确定，你就不能对网页做任何的操作。没错，这个“咚”的小窗口就是 alert 干的。

下面的代码是一段使用 alert 的实例。

```
<script type="text/JavaScript">
     alert("我是菜鸟我怕谁");
</script>
```

9.2 确认(confirm)

确认框用于让用户选择某一个问题是否符合实际情况。来看下面的代码：我们用 confirm("你是菜鸟吗？")向访客提问，变量 r 则保存了访客的回应，它只可能有两种取值：true 或 false。没错，它是一个布尔值。confirm 后面的语句则是我们对访客回答做出的不同回应。

```
<script type="text/JavaScript">
     var r=confirm("你是菜鸟吗");
     if (r==true)
     {
     document.write("彼此彼此");
     }
     else
     {
     document.write("佩服佩服");
     }
</script>
```

9.3 提问(prompt)

prompt 和 confirm 类似，不过它允许访客随意输入回答。我们来修改一下之前 switch 的例子，我们根据分数来做出不同的评价，不过那段程序并不完整，它根本就没问我们分数，而是假设我得了 65 分。这太不公平了。现在我么就可以用 prompt 来向访客提问，用 score 存储用户输入的回答，其余的事情就都由后面的 switch 来完成了。

```
<script type="text/JavaScript">
 function judge() {
 var score;//分数
 var degree;//分数等级
 score = prompt("你的分数是多少？")
 if (score > 100){
 degree = '耍我？100 分满！';
```

```
}
else{
switch (parseInt(score / 10)) {
case 0:
case 1:
case 2:
case 3:
case 4:
case 5:
degree = "恭喜你，又挂了！";
break;
case 6:
degree = "勉强及格";
break;
case 7:
degree = "凑合，凑合"
break;
case 8:
degree = "8 错，8 错";
break;
case 9:
case 10:
degree = "高手高手，佩服佩服";
}//end of switch
}//end of else
alert(degree);
}
</script>
```

prompt( )方法除了可以给出提示信息外，还可以给出初始值，格式如下：

prompt(“提示信息”，“初值”)

10 JavaScript 函数

通常情况下，函数是完成特定功能的一段代码。把一段完成特定功能的代码块放到一个函数里，以后就可以调用这个函数啦，这样就省去了重复输入大量代码的麻烦。还记得我们上一节所使用的 alert 吗？其实它就是一个函数，他是 JavaScript 提供的函数。

一个函数的作用就是完成一项特定的任务。如果没有函数时，完成一项任务可能需要五行、十行、甚至更多的代码。每次需要完成这个任务的时候都重写一遍代码显然不是一个好主意。这是我们就可以编写一个函数来完成这个任务，以后只要调用这个函数就可以了。

10.1 定义一个函数

说了这么多，那么究竟如何定义一个函数呢？看看下面的格式：

```
function 函数名()
{
        函数代码;
}
```

函数由关键字 Function 定义，把“函数名”替换为你想要的名字，把“代码”替换为完成特定功能的代码，函数就定义好了。了解了如何定义函数，我们就来自己编写一个实现两数相加的简单函数吧。

首先给函数起一个有意义的名字：“add2”，它的代码如下：

```
function add2(){
 sum = 1 + 1;
 alert(sum);
}
```

10.2 函数的调用

函数定义好了，如何调用呢？

其实可以通过很多种方法调用上面的函数，我们这里使用最简单的函数调用方式——按钮的点击事件，JavaScript 事件会在后面介绍。试着点击下面的按钮调用定义好的函数：

```
<html>
<head>
<script language=javascript>
     function add2(){
     sum = 1 + 1;
     alert(sum);
   }
</SCRIPT>
</head>
<body>
<form>
<input type="button" value="click it" ONCLICK=" add2 ()">
</form>
</body>
```

```
</html>
```

通过 button 按钮的鼠标单击事件 onclick 调用 add2()函数。

10.3 带参数的函数

上述 Add2()函数不能实现任意指定两数相加。其实，函数的定义可以是下面的格式：

```
function(参数 1,参数 2,参数 3){
        部分函数代码……
……
}
```

按照这个格式，我们的函数应该写成：

```
function add2(x,y){
 sum = x + y;
 alert(sum);
}
```

x 和 y 则是我们函数的两个参数，调用函数的时候，我们就可以通过这两个参数把两个加数传递给函数了。例如，add2(3，4)会求 3+4 的和，add2(56,65)则会求出 56 和 65 的和。

“再等等!这函数没有用啊,这里只是把结果 alert 出来,我想对结果做些处理怎么办啊？”

10.4 带返回值的函数

好吧，我们把 alert(sum)一行改成下面的代码：

```
return sum;
```

return 后面的值叫做返回值。使用下面的语句调用函数就可以将这个返回值存储在变量中了。

```
result = add2(3,4);
```

该语句执行后，result 变量中的值为 7。值得说明的是，我们的函数中，参数和返回值

都是数字，其实它们也可以是字符串等其它类型。

11 JavaScript 事件

我们之前提到过函数的调用。函数定义之后，默认是不会执行的，这时候就需要一些事件来触发这个函数的执行。

JavaScript 很多有很多事件，例如鼠标的点击、移动，网页的载入和关闭。我们先来定义一个函数，再看几个事件的实例。

统一的示例函数：

```
<script type="text/JavaScript">
function displaymessage()
{
alert("我是菜鸟我怕谁!");
}
</script>
```

函数的事件很简单，只是显示一条消息。

11.1 点击事件

使用点击事件调用，需要给元素设置 onclick 属性。示例代码如下：

```
<button value="点击提交" onclick="displaymessage()">onclick 调用函数</button>
```

由于设置了 onclick="displaymessage()"，因此点击按钮则会调用函数。

11.2 鼠标经过、移出事件

使用鼠标经过事件调用函数的代码如下：

```
<button value="点击提交" onmouseover="displaymessage()">鼠标滑过调用函数</button>
```

当鼠标经过按钮时，触发 onmouseover 事件，将会调用函数 displaymessage()。

使用鼠标移出事件调用函数的代码如下：

```
<button value="点击提交" onmouseout="displaymessage()">鼠标移出调用函数</button>
```

把鼠标移动到这个按钮里面，当再移动出去时，触发 onmouseout 事件，将会调用函数 displaymessage()。

11.3 更多事件

JavaScript 中还有很多事件，完整的列表可以看看 http://www.w3pop.com/learn/view/p/3/o/0/doc/jsref_events/。

下面的列表列举了可以插入 HTML 标签中来定义事件动作的属性，具体的用法请参考上面的网站。

属性	事件发生时机
onabort	图片下载被打断时
onblur	元素失去焦点时
onchange	框内容改变时
onclick	鼠标点击一个对象时
ondblclick	鼠标双击一个对象时
onerror	当加载文档或图片时发生错误时
onfocus	当元素获取焦点时
onkeydown	按下键盘按键时
onkeypress	按下或按住键盘按键时
onkeyup	放开键盘按键时
onload	页面或图片加载完成时
onmousedown	鼠标被按下时
onmousemove	鼠标被移动时
onmouseout	鼠标离开元素时
onmouseover	鼠标经过元素时
onmouseup	释放鼠标按键时
onreset	重新点击鼠标按键时
onresize	当窗口或框架被重新定义尺寸时
onselect	文本被选择时
onsubmit	点击提交按钮时
onunload	用户离开页面时

12 JavaScript 特殊字符

JavaScript 中有一些特殊的字符，如果想要在字符串中使用，需要加上一个反斜线。

我们在之前已经看过好多应用字符串的例子，如果现在想输出内容将是：小明说：″我是菜鸟我怕谁？″。

该怎么办呢？双引号被当作 javascript 标记字符串开始和结束的符号，我们怎么能在字符串里引入双引号呢?看看下面这个例子。

```
document.write("小明说：\"我是菜鸟我怕谁？\"。");
```

我们在双引号前面加上一个反斜线就可以了。在 JavaScript 中有很多这样具有功能的特殊字符，如果需要把它们插入字符串都需要在前面加上一个反斜线"\"。

特殊字符列表

JavaScript 特殊字符代码	
代码	**输出**
\’	单引号
\″	双引号
\&	与符号
\\	反斜线
\n	换行
\r	回车
\t	制表符
\b	后退
\f	走纸换页

13 JavaScript 指导准则

（1）在书写 JavaScript 的过程中，应该记住它是区分大小写的，变量名和函数名都区分大小写。比如：

```
Var A=1;
Var a=2;
```

A 和 a 是两个不同的变量。

（2）代码中多余的空格是被忽略的。例如下面两个语句

```
a=b+c;
```

```
a = b + c ;
```

他们是等价的。

（3）”\”表示本行未完

前面已经说过，浏览器读到一行末尾会自动判定本行已经结束，不过我们可以通过在行末添加一个“\”来告诉浏览器本行没有结束。

```
document.write("Hello \
  World!")
```

和

```
document.write("我是菜鸟我怕谁!")
```

是完全一样的。

14 JavaScript 对象化编程

JavaScript 是使用“对象化编程”的，或者叫“面向对象编程”的。所谓“对象化编程”，意思是把 JavaScript 能涉及的范围划分成大大小小的对象，对象下面还继续划分对象直至非常详细为止，所有的编程都以对象为出发点，基于对象。小到一个变量，大到网页文档、窗口甚至屏幕，都是对象。

JavaScript **对象**是可以是一段文字、一幅图片、一个表单（Form）等等。每个对象有它自己的**属性**、**方法**和**事件**。对象的**属性**是反映该对象某些特定的性质的，例如：字符串的长度、图像的长宽、文字框（Textbox）里的文字等等；对象的**方法**能对该对象做一些事情，例如，表单的“提交”(Submit)，窗口的“滚动”(Scrolling)等等；而对象的**事件**就能响应发生在对象上的事情，例如提交表单产生表单的“提交事件”，点击连接产生的“点击事件”。不是所有的对象都有以上三个性质，有些没有事件，有些只有属性。引用对象的任一“性质”用“<对象名>.<性质名>”这种方法。

JavaScript 对象有：基本对象、全局对象、文档对象。下面我们一一介绍。

14.1 基本对象

（1）String 字符串对象

我们在之前的学习中已经就在使用字符串对象对象了，声明一个字符串对象的方法就是直接赋值。比如：

```
var s = "我有个 7 个字符";
```

定义了 s 这个字符串之后，我们就有了一个字符串对象，我们可以访问它的属性，使用它的方法。

属性

length 用法：<字符串对象>.length；返回该字符串的长度。

方法

charAt() 用法：<字符串对象>.charAt(<位置>)；返回该字符串位于第<位置>位的单个字符。注意：字符串中的一个字符是第 0 位的，第二个才是第 1 位的，最后一个字符是第 length - 1 位的。

charCodeAt() 用法：<字符串对象>.charCodeAt(<位置>)；返回该字符串位于第<位置>位的单个字符的 ASCII 码。

fromCharCode() 用法：String.fromCharCode(a, b, c...)；返回一个字符串，该字符串每个字符的 ASCII 码由 a, b, c... 等来确定。

indexOf() 用法：<字符串对象>.indexOf(<另一个字符串对象>[, <起始位置>])；该方法从<字符串对象>中查找<另一个字符串对象>（如果给出<起始位置>就忽略之前的位置），如果找到了，就返回它的位置，没有找到就返回“-1”。所有的“位置”都是从零开始的。

lastIndexOf() 用法：<字符串对象>.lastIndexOf(<另一个字符串对象>[, <起始位置>])；跟 indexOf() 相似，不过是从后边开始找。

split() 用法：<字符串对象>.split(<分隔符字符>)；返回一个数组，该数组是从<字符串对象>中分离开来的，<分隔符字符>决定了分离的地方，它本身不会包含在所返回的数组中。例如：'1&2&345&678'.split('&')返回数组：1,2,345,678。关于数组，我们等一下就讨论。

substring() 用法：<字符串对象>.substring(<始>[, <终>])；返回原字符串的子字符串，该字符串是原字符串从<始>位置到<终>位置的前一位置的一段。<终> - <始> = 返回字符串的长度（length）。如果没有指定<终>或指定得超过字符串长度，则子字符串从<始>位置一直取到原字符串尾。如果所指定的位置不能返回字符串，则返回空字符串。

substr() 用法：<字符串对象>.substr(<始>[, <长>])；返回原字符串的子字符串，该字符串是原字符串从<始>位置开始，长度为<长>的一段。如果没有指定<长>或指定得超过字符串长度，则子字符串从<始>位置一直取到原字符串尾。如果所指定的位置不能返回字符串，则返回空字符串。

toLowerCase() 用法：<字符串对象>.toLowerCase()；返回把原字符串所有大写字母都变成小写的字符串。

toUpperCase() 用法：<字符串对象>.toUpperCase()；返回把原字符串所有小写字母都变成大写的字符串。

（2）Array 数组对象

数组对象是一个对象的集合，里边的对象可以是不同类型的。数组的每一个成员对象都

有一个“下标”，用来表示它在数组中的位置，是从零开始的。

数组的定义方法：

var <数组名> = new Array();

这样就定义了一个空数组。以后要添加数组元素，就用：

<数组名>[<下标>] = ...;

数组的下标用方括号括起来。

如果想在定义数组的时候直接初始化数据，请用：

var <数组名> = new Array(<元素 1>, <元素 2>, <元素 3>...);

例如：

```
var myArray = new Array(1, 4.5, 'Hi');
```

定义了一个数组 myArray，里边的元素是：myArray[0] = 1; myArray[1] = 4.5; myArray[2] = 'Hi'。

定义时指定有 n 个空元素的数组，请用：

var a=new Array(n);

属性

length 用法：<数组对象>.length；返回：数组的长度，即数组里有多少个元素。它等于数组里最后一个元素的下标加一。

方法

join() 用法：<数组对象>.join(<分隔符>)；返回一个字符串，该字符串把数组中的各个元素串起来，用<分隔符>置于元素与元素之间。这个方法不影响数组原本的内容。

reverse() 用法：<数组对象>.reverse()；使数组中的元素顺序反过来。如果对数组[1, 2, 3]使用这个方法，它将使数组变成：[3, 2, 1]。

slice() 用法：<数组对象>.slice(<始>[, <终>])；返回一个数组，该数组是原数组的子集，始于<始>，终于<终>。如果不给出<终>，则子集一直取到原数组的结尾。

sort() 用法：<数组对象>.sort([<方法函数>])；使数组中的元素按照一定的顺序排列。如果不指定<方法函数>，则按字母顺序排列。在这种情况下，80 是比 9 排得前的。如果指定<方法函数>，则按<方法函数>所指定的排序方法排序。<方法函数>比较难讲述，这里只将一些有用的<方法函数>介绍给大家。

按升序排列数字：

```
function sortMethod(a, b) {
 return a - b;
}
myArray.sort(sortMethod);
```

按降序排列数字：把上面的“a - b”该成“b - a”。

（3）Math “数学”对象

Math 对象，提供对数据的数学计算。下面所提到的属性和方法，不再详细说明“用法”，大家在使用的时候记住用“Math.<名>”这种格式。

属性

E 返回常数 e (2.718281828...)。
LN2 返回 2 的自然对数 (ln 2)。
LN10 返回 10 的自然对数 (ln 10)。
LOG2E 返回以 2 为低的 e 的对数 ($\log_2 e$)。
LOG10E 返回以 10 为低的 e 的对数 ($\log_{10} e$)。
PI 返回π（3.1415926535...)。
SQRT1_2 返回 1/2 的平方根。
SQRT2 返回 2 的平方根。

方法

abs(x) 返回 x 的绝对值。
acos(x) 返回 x 的反余弦值（余弦值等于 x 的角度），用弧度表示。
asin(x) 返回 x 的反正弦值。
atan(x) 返回 x 的反正切值。
atan2(x, y) 返回复平面内点(x, y)对应的复数的幅角，用弧度表示，其值在-π 到π之间。
ceil(x) 返回大于等于 x 的最小整数。
cos(x) 返回 x 的余弦。
exp(x) 返回 e 的 x 次幂 (e^x)。
floor(x) 返回小于等于 x 的最大整数。
log(x) 返回 x 的自然对数 (ln x)。
max(a, b) 返回 a, b 中较大的数。
min(a, b) 返回 a, b 中较小的数。
pow(n, m) 返回 n 的 m 次幂 (n^m)。
random() 返回大于 0 小于 1 的一个随机数。
round(x) 返回 x 四舍五入后的值。
sin(x) 返回 x 的正弦。
sqrt(x) 返回 x 的平方根。
tan(x) 返回 x 的正切。

（4）Date 对象

Date 日期对象。这个对象可以储存任意一个日期，从 0001 年到 9999 年，并且可以精确到毫秒数（1/1000 秒）。

定义一个日期对象：

```
var today = new Date();
```

这个方法使 d 成为日期对象，并且已有初始值：当前时间。如果要自定初始值，可以用下列方法：

```
var d = new Date(99, 10, 1);  //99 年 10 月 1 日
var d = new Date('Oct 1, 1999'); //99 年 10 月 1 日
```

最好的方法就是用下面介绍的“方法”来严格的定义时间。

方法

以下有很多 getXXX()、setXXX（）这样的方法，getXXX()是获得某个数值，而 setXXX（）是设定某个数值。

如无说明，方法的使用格式为：“<对象>.<方法>”，下同。

get/setFullYear() 返回/设置年份，用四位数表示。如果使用“x.setFullYear(99)”，则年份被设定为 0099 年。
get/setYear() 返回/设置年份，用两位数表示。设定的时候浏览器自动加上“19”开头，故使用“x.setYear(00)”把年份设定为 1900 年。
get/setMonth() 返回/设置月份，0 表示 1 月。
get/setDate() 返回/设置日期。
get/setDay() 返回/设置星期，0 表示星期天。
get/setHours() 返回/设置小时数，24 小时制。
get/setMinutes() 返回/设置分钟数。
get/setSeconds() 返回/设置秒钟数。
get/setMilliseconds() 返回/设置毫秒数。
get/setTime() 返回/设置时间，该时间就是日期对象的内部处理方法：从 1970 年 1 月 1 日零时正开始计算到日期对象所指的日期的毫秒数。如果要使某日期对象所指的时间推迟 1 小时，就用：“x.setTime(x.getTime() + 60 * 60 * 1000);”（一小时 60 分，一分 60 秒，一秒 1000 毫秒）。
getTimezoneOffset() 返回日期对象采用的时区与格林威治时间所差的分钟数。在格林威治东方的市区，该值为负，例如：中国时区（GMT+0800）返回“-480”。
toString() 返回一个字符串，描述日期对象所指的日期。这个字符串的格式类似于：“Fri

Jul 21 15:43:46 UTC+0800 2000”。
toLocaleString() 返回一个字符串，描述日期对象所指的日期，用本地时间表示格式。如：“2000-07-21 15:43:46”。
toGMTString() 返回一个字符串，描述日期对象所指的日期，用 GMT 格式。
toUTCString() 返回一个字符串，描述日期对象所指的日期，用 UTC 格式。
parse() 用法：Date.parse(<日期对象>)；返回该日期对象的内部表达方式。

下面例子显示当前日期：

```
<html>
<body>
<script language="JavaScript">
today = new Date();
var day; var date;
if(today.getDay()==0) day = "星期日"
if(today.getDay()==1) day = "星期一"
if(today.getDay()==2) day = "星期二"
if(today.getDay()==3) day = "星期三"
if(today.getDay()==4) day = "星期四"
if(today.getDay()==5) day = "星期五"
if(today.getDay()==6) day = "星期六"
   date = "今天是" + (today.getYear()) + "年" + (today.getMonth() + 1 ) + "月
" + today.getDate() + " 日 " + day +"";
document.write(date);
</script>
</body>
</html>
```

14.2 全局对象

全局对象就是一些全局函数，他们可以直接用，这里对它们进行简要的介绍。

eval() 把括号内的字符串当作标准语句或表达式来运行。

```
b = "2+5*2";
var a=eval(b);
```

a 的值为 12。

isNaN() 如果括号内的值是“NaN（不是数字）”,则返回 true 否则返回 false。

parseInt() 返回把括号内的内容转换成整数之后的值。如果括号内是字符串，则字符串开头的数字部分被转换成整数，如果以字母开头，则返回“NaN”。

parseFloat() 返回把括号内的字符串转换成浮点数之后的值，字符串开头的数字部分被转换成浮点数，如果以字母开头，则返回“NaN”。

toString() 用法：<对象>.toString()；把对象转换成字符串。如果在括号中指定一个数值，则转换过程中所有数值转换成特定进制。

escape() 返回括号中的字符串经过编码后的新字符串。该编码应用于 URL，也就是把空格写成“%20”这种格式。“+”不被编码，如果要“+”也被编码，请用：escape('...', 1)。

unescape() 是 escape() 的反过程。解编括号中字符串成为一般字符串。

14.3 文档对象

文档对象是指在网页文档里划分出来的对象。在 JavaScript 能够涉及的范围内有如下几个“大”对象：window, document, location, navigator, screen, history 等。下面是一个文档对象树。

要引用某个对象，就要把父级的对象都列出来。例如，要引用某表单“MyForm”的某文字框“UserName”，就要用“document. MyForm. UserName”。

引用 Form 下的表单元素对象不使用名称，比如 Button，而是通过对象的 ID 或 Name 进行引用，或使用它所属的对象数组。比如：

```
<input id="UserName" type="text" />
        …
var name = document.getElementById("UserName");//通过 id 获取值
```

- 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
        - applets[] Java 小程序对象
        - embeds[] 插件对象
        - 表单对象

- forms[]; Form
  - 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 选择区（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

（1）navigator

navigator 浏览器对象 反映了当前使用的浏览器的资料。

属性

appCodeName 返回浏览器的“码名”，流行的 IE 和 NN 都返回 'Mozilla'。
appName 返回浏览器名。IE 返回 'Microsoft Internet Explorer'，NN 返回 'Netscape'。
appVersion 返回浏览器版本，包括了大版本号、小版本号、语言、操作平台等信息。
platform 返回浏览器的操作平台，对于 Windows 9x 上的浏览器，返回 'Win32'（大小写可能有差异）。
userAgent 返回以上全部信息。例如，IE5.01 返回 'Mozilla/4.0 (compatible; MSIE 5.01; Windows 98)'。
javaEnabled() 返回一个布尔值，代表当前浏览器允许不允许 Java。

（2）screen

screen 屏幕对象 反映了当前用户的屏幕设置。

属性

width 返回屏幕的宽度（像素数）。
height 返回屏幕的高度。
availWidth 返回屏幕的可用宽度（除去了一些不自动隐藏的类似任务栏的东西所占用的宽度）。
availHeight 返回屏幕的可用高度。
colorDepth 返回当前颜色设置所用的位数 - 1：黑白；8：256 色；16：增强色；24/32：真彩色

（3）window

window 窗口对象是最大的对象，它描述的是一个浏览器窗口。一般要引用它的属性和方法时，不需要用“window.xxx”这种形式，而直接使用“xxx”。一个框架页面也是一个窗口。

属性

name 窗口的名称，由打开它的连接（）或框架页（<frame name="...">）或某一个窗口调用的 open() 方法决定。一般我们不会用这个属性。
status 指窗口下方的“状态栏”所显示的内容。通过对 status 赋值，可以改变状态栏的显示。
opener 用法：window.opener；返回打开本窗口的窗口对象。注意：返回的是一个窗口对象。如果窗口不是由其他窗口打开的，在 Netscape 中这个属性返回 null；在 IE 中返回“未定义”(undefined)。undefined 在一定程度上等于 null。注意：undefined 不是 JavaScript 常数，如果你企图使用“undefined”，那就真的返回“未定义”了。
self 指窗口本身，它返回的对象跟 window 对象是一模一样的。最常用的是“self.close()”，放在<a>标记中：“a href="javascript:self.close()">关闭窗口”。
parent 返回窗口所属的框架页对象。
top 返回占据整个浏览器窗口的最顶端的框架页对象。
history 历史对象，见下。
location 地址对象，见下。
document 文档对象，见下。

方法

open() 打开一个窗口。
用法：open(<URL 字符串>, <窗口名称字符串>, <参数字符串>);
<URL 字符串>：描述所打开的窗口打开哪一个网页。如果留空（''），则不打开任何网页。
<窗口名称字符串>：描述被打开的窗口的名称(window.name)，可以使用'_top'、'_blank'等内建名称。这里的名称跟“<a href="..." target="...">”里的“target”属性是一样的。
<参数字符串>：描述被打开的窗口的样式。如果只需要打开一个普通窗口，该字符串留空（''），如果要指定样式，就在字符串里写上一到多个参数，参数之间用逗号隔开。

例：打开一个 400 x 100 的干净的窗口：

```
open('','_blank','width=400,height=100,menubar=no,toolbar=no,
location=no,directories=no,status=no,scrollbars=yes,resizable=yes')
```

参数

top=#	窗口顶部离开屏幕顶部的像素数
left=#	窗口左端离开屏幕左端的像素数
width=#	窗口的宽度
height=#	窗口的高度
menubar=...	窗口有没有菜单，取值 yes 或 no

toolbar=...	窗口有没有工具条，取值 yes 或 no
location=...	窗口有没有地址栏，取值 yes 或 no
directories=...	窗口有没有连接区，取值 yes 或 no
scrollbars=...	窗口有没有滚动条，取值 yes 或 no
status=...	窗口有没有状态栏，取值 yes 或 no
resizable=...	窗口给不给调整大小，取值 yes 或 no
fullscreen=	窗口是否全屏，取值 yes 或 no

open() 方法有返回值，返回的就是它打开的窗口对象。所以，

```
var newWindow = open('','_blank');
```

这样把一个新窗口赋值到“newWindow”变量中，以后通过“newWindow”变量就可以控制窗口了。

close()关闭一个已打开的窗口。
用法：window.close() 或 self.close()：关闭本窗口；
<窗口对象>.close()：关闭指定的窗口。
如果该窗口有状态栏，调用该方法后浏览器会警告：“网页正在试图关闭窗口，是否关闭？”然后等待用户选择是否；如果没有状态栏，调用该方法将直接关闭窗口。
blur() 使焦点从窗口移走，窗口变为“非活动窗口”。
focus() 是窗口获得焦点，变为“活动窗口”。
scrollTo() 用法:[<窗口对象>.]scrollTo(x, y);使窗口滚动,使文档从左上角数起的(x, y)点滚动到窗口的左上角。
scrollBy() 用法:[<窗口对象>.]scrollBy(deltaX, deltaY);使窗口向右滚动 deltaX 像素，向下滚动 deltaY 像素。如果取负值，则向相反的方向滚动。
resizeTo() 用法:[<窗口对象>.]resizeTo(width, height);使窗口调整大小到宽 width 像素，高 height 像素。
resizeBy() 用法:[<窗口对象>.]resizeBy(deltaWidth, deltaHeight);使窗口调整大小，宽增大 deltaWidth 像素，高增大 deltaHeight 像素。如果取负值，则减少。
alert() 用法：alert(<字符串>)；弹出一个只包含“确定”按钮的对话框，显示<字符串>的内容，整个文档的读取、Script 的运行都会暂停，直到用户按下“确定”。
confirm() 用法：confirm(<字符串>)；弹出一个包含“确定”和“取消”按钮的对话框，显示<字符串>的内容，要求用户做出选择，整个文档的读取、Script 的运行都会暂停。如果用户按下“确定”，则返回 true 值，如果按下“取消”，则返回 false 值。
prompt() 用法：prompt(<字符串>[, <初始值>])；弹出一个包含“确认”“取消”和一个文本框的对话框，显示<字符串>的内容，要求用户在文本框输入一些数据，整个文档的读取、Script 的运行都会暂停。如果用户按下“确认”，则返回文本框里已有的内容，如果用户按下“取消”，则返回 null 值。如果指定<初始值>，则文本框里会有默认值。

setTimeout()和 setInterval()的使用

这两个方法都可以用来实现在一个固定时间段之后去执行 JavaScript。不过两者各有各

的应用场景。

实际上，setTimeout 和 setInterval 的语法相同。它们都有两个参数，一个是将要执行的代码字符串，还有一个是以毫秒为单位的时间间隔，当过了那个时间段之后就将执行那段代码。

不过这两个函数还是有区别的，setInterval 在执行完一次代码之后，经过了那个固定的时间间隔，它还会自动重复执行代码，而 setTimeout 只执行一次那段代码。

虽然表面上看来 setTimeout 只能应用在 on-off 方式的动作上，不过可以通过创建一个函数循环重复调用 setTimeout，以实现重复的操作：

```
showTime();
function showTime()
{
      var today = new Date();
      alert("The time is: " + today.toString());
      setTimeout("showTime()", 5000);
}
```

一旦调用了这个函数，那么就会每隔 5 秒钟就显示一次时间。如果使用 setInterval，则相应的代码如下所示：

```
setInterval("showTime()", 5000);
function showTime()
{
      var today = new Date();
      alert("The time is: " + today.toString());
}
```

这两种方法可能看起来非常像，而且显示的结果也会很相似，不过两者的最大区别就是，setTimeout 方法不会每隔 5 秒钟就执行一次 showTime 函数，它是在每次调用 setTimeout 后过 5 秒钟再去执行 showTime 函数。这意味着如果 showTime 函数的主体部分需要 2 秒钟执行完，那么整个函数则要每 7 秒钟才执行一次。而 setInterval 却没有被自己所调用的函数所束缚，它只是简单地每隔一定时间就重复执行一次那个函数。

如果要求在每隔一个固定的时间间隔后就精确地执行某动作，那么最好使用 setInterval，而如果不想由于连续调用产生互相干扰的问题，尤其是每次函数的调用需要繁重的计算以及很长的处理时间，那么最好使用 setTimeout。

用 setInterval 命令来创建的对象，可以用 clearInterval() 命令来终止。比如：

```
var MyMar=setInterval(showTime(),speed);
clearInterval(MyMar);
```

（4）history

history 历史对象指浏览器的浏览历史。

属性

length 历史的项数。JavaScript 所能管到的历史被限制在用浏览器的“前进”“后退”键可以去到的范围。本属性返回的是“前进”和“后退”两个按键之下包含的地址数的和。

方法

back() 后退，跟按下“后退”键是等效的。
forward() 前进，跟按下“前进”键是等效的。
go() 用法：history.go(x)；在历史的范围内去到指定的一个地址。如果 x＜0，则后退 x 个地址，如果 x＞0，则前进 x 个地址，如果 x == 0，则刷新现在打开的网页。history.go(0) 跟 location.reload() 是等效的。

（5）location

location 地址对象描述的是某一个窗口对象所打开的地址。要表示当前窗口的地址，只需要使用“location”就行了；若要表示某一个窗口的地址，就使用“<窗口对象>.location”。

注意 属于不同协议或不同主机的两个地址之间不能互相引用对方的 location 对象，这是出于安全性的需要。例如，当前窗口打开的是“www.a.com”下面的某一页，另外一个窗口（对象名为：bWindow）打开的是“www.b.com”的网页。如果在当前窗口使用“bWindow.location”，就会出错：“没有权限”。这个错误是不能用错误处理程序（Event Handler，参阅 onerror 事件)来接收处理的。

属性

protocol 返回地址的协议，取值为 'http:','https:','file:' 等等。
hostname 返回地址的主机名，例如，一个“http://www.microsoft.com/china/”的地址，location.hostname == 'www.microsoft.com'。
port 返回地址的端口号，一般 http 的端口号是 '80'。
host 返回主机名和端口号，如：'www.a.com:8080'。
pathname 返回路径名，如“http://www.a.com/b/c.html”，location.pathname == 'b/c.html'。
hash 返回“#”以及以后的内容，如“http://www.a.com/b/c.html#chapter4”，location.hash == '#chapter4'；如果地址里没有“#”，则返回空字符串。
search 返回“?”以及以后的内容，如“http://www.a.com/b/c.asp?selection=3&jumpto=4”，location.search =='?selection=3&jumpto=4'；如果地址里没有“?”，则返回空字符串。
href 返回以上全部内容，也就是说，返回整个地址。在浏览器的地址栏上怎么显示它就怎么返回。如果想一个窗口对象打开某地址，可以使用“location.href = '...'”，也可以直接用“location = '...'”来达到此目的。

方法

reload() 相当于按浏览器上的“刷新”(IE)或“Reload”(Netscape)键。
replace() 打开一个 URL，并取代历史对象中当前位置的地址。用这个方法打开一个 URL 后，按下浏览器的“后退”键将不能返回到刚才的页面。

（6）document

document 文档对象 描述当前窗口或指定窗口对象的文档。它包含了文档从<head>到</body>的内容。

用法：document （当前窗口）

或 <窗口对象>.document （指定窗口）

属性

lastModified 当前文档的最后修改日期，是一个 Date 对象。
referrer 如果当前文档是通过点击连接打开的，则 referrer 返回原来的 URL。
title 指<head>标记里用<title>...</title>定义的文字。在 Netscape 里本属性不接受赋值。
fgColor 指<body>标记的 text 属性所表示的文本颜色。
bgColor 指<body>标记的 bgcolor 属性所表示的背景颜色。
linkColor 指<body>标记的 link 属性所表示的连接颜色。
alinkColor 指<body>标记的 alink 属性所表示的活动连接颜色。
vlinkColor 指<body>标记的 vlink 属性所表示的已访问连接颜色。

方法

open() 打开文档以便 JavaScript 能向文档的当前位置（指插入 JavaScript 的位置）写入数据。通常不需要用这个方法，在需要的时候 JavaScript 自动调用。
write();writeln() 向文档写入数据，所写入的会当成标准文档 HTML 来处理。writeln() 与 write() 的不同点在于，writeln() 在写入数据以后会加一个换行。这个换行只是在 HTML 中换行，具体情况能不能够是显示出来的文字换行，要看插入 JavaScript 的位置而定。如在<pre>标记中插入，这个换行也会体现在文档中。
clear() 清空当前文档。
close() 关闭文档，停止写入数据。如果用了 write[ln]() 或 clear() 方法，就一定要用 close() 方法来保证所做的更改能够显示出来。如果文档还没有完全读取，也就是说，JavaScript 是插在文档中的，那就不必使用该方法。

现在我们已经拥有足够的知识来做以下这个很多网站都有的弹出式更新通知了。

```
<script language="JavaScript">
<!--
var whatsNew = open('','_blank','top=50,left=50,width=200,height=300,' +
                                'menubar=no,toolbar=no,directories=no,locatio
n=no,' +
                                'status=no,resizable=no,scrollbars=yes');
whatsNew.document.write('<center><b>更新通知</b></center>');
whatsNew.document.write('<p>最后更新日期：00.08.01');
whatsNew.document.write('<p>00.08.01：增加了“我的最爱”栏目。');
```

```
whatsNew.document.write('<p align="right">' +
                              '<a href="javascript:self.close()">关闭窗口
</a>');
whatsNew.document.close();
-->
</script>
```

当然也可以先写好一个 HTML 文件，在 open() 方法中直接 load 这个文件。

（7）anchors[]; links[]; Link

anchors[]; links[]; Link 连接对象。

用法：document.anchors[[x]]; document.links[[x]]; <anchorId>; <linkId>

document.anchors 是一个数组，包含了文档中所有锚标记（包含 name 属性的<a>标记），按照在文档中的次序，从 0 开始给每个锚标记定义了一个下标。

document.links 也是一个数组，包含了文档中所有连接标记（包含 href 属性的<a>标记和<map>标记段里的<area>标记），按照在文档中的次序，从 0 开始给每个连接标记定义了一个下标。

如果一个<a>标记既有 name 属性，又有 href 属性，则它既是一个 Anchor 对象，又是一个 Link 对象。

在 IE 中,如果在<a>标记中添加“id="..."”属性,则这个<a>对象被赋予一个标识(ID),调用这个对象的时候只需要使用“<id>”就行了。很多文档部件都可以用这个方法来赋予 ID,但要注意不能有两个 ID 相同。

anchors 和 links 作为数组，有数组的属性和方法。单个 Anchor 对象没有属性；单个 Link 对象的属性见下。

属性

protocol; hostname; port; host; pathname; hash; search; href 与 location 对象相同。

target 返回/指定连接的目标窗口（字符串），与<a>标记里的 target 属性是一样的。

15 JavaScript 实例

15.1 父窗口获取弹出子窗口文本框的值

```
＜script defer＞
function browser.onclick(){
var nW = window.open("","","width=170,height=200");
nW.moveTo(300,100);
nW.document.write("＜script defer＞\r"+
                  "var m=true;\r"+
                "function insert.onclick(){\r"+
                  "if(m){\r"+
                        "opener.atext.value=btext.value;\rm=false;"+
                        "}else{\r"+
                        "opener.atext.value+='\-'+btext.value;\r"+
                "}\r}\r"+
                "＜\/script＞\r"+
                   "＜input name=btext value=111111 size=10 style='BACKGROUND:
#ffffff; BORDER-BOTTOM: #0A3EAE 1px solid; BORDER-LEFT: #0A3EAE 1px solid;
BORDER-RIGHT: #0A3EAE 1px solid; BORDER-TOP: #0A3EAE 1px solid'＞\r"+
                          "＜input type=button value=选择 name=insert
style='background-color: #EBF5FC; border: 1px solid #0099CC'＞");
}
＜/script＞
＜textarea name="atext" cols="60" rows="4" ＞＜/textarea＞
＜input type=button value=选择参数 name=browser style="background-color: #EBF5FC;
border: 1px solid #0099CC"＞
```