Javascript 简介

学习 Javascript 之前

学习 Javascript 之前，请先具备 HTML 和 CSS 的知识。参见 HTML 教程，CSS 教程。

Javascript 简介

Javascript 是一种解释性的，基于对象的脚本语言(an interpreted, object-based scripting language)。

HTML 网页在互动性方面能力较弱，例如下拉菜单，就是用户点击某一菜单项时，自动会出现该菜单项的所有子菜单，用纯 HTML 网页无法实现；又如验证 HTML 表单(Form)提交信息的有效性，用户名不能为空，密码不能少于4位，邮政编码只能是数字之类，用纯 HTML 网页也无法实现。要实现这些功能，就需要用到 Javascript。

Javascript 是一种脚本语言，比 HTML 要复杂。不过即便你先前不懂编程，也不用担心，因为 Javascript 写的程序都是以源代码的形式出现的，也就是说你在一个网页里看到一段比较好的 Javascript 代码，恰好你也用得上，就可以直接拷贝，然后放到你的网页中去。正因为可以借鉴、参考优秀网页的代码，所以让 Javascript 本身也变得非常受欢迎，从而被广泛应用。原来不懂编程的人，多参考 Javascript 示例代码，也能很快上手。

Javascript 主要是基于客户端运行的，用户点击带有 Javascript 的网页，网页里的 Javascript 就传到浏览器，由浏览器对此作处理。前面提到的下拉菜单、验证表单有效性等大量互动性功能，都是在客户端完成的，不需要和 Web Server 发生任何数据交换， 因此，不会增加 Web Server 的负担。

几乎所有浏览器都支持 Javascript，如 Internet Explorer(IE)，Firefox， Netscape， Mozilla，Opera 等。

简单的 Javascript 入门示例

我们先来看一个最简单的例子，代码如下：

```
<html>
<head><title>一个最简单的 Javascript 示例(仅使用了 document.write)</title></head>
<body>
<script type="text/javascript">
   document.write("Hello, World!");
</script>
</body>
</html>
```

演示示例

```
<html>
<head><title>一个最简单的 Javascript 示例(仅使用了 document.write)</title></head>
<body>
<script type="text/javascript">
document.write("Hello, World!");
</script>
</body>
</html>
```

在 HTML 网页里插入 Javascript 语句，应使用 HTML 的<script>。<script>这个 tag 有个属性叫 type，type="text/javascript"表示插入<script></script>其中的为 Javascript 语句。

上面的例子中，使用了 document.wirte，这是 Javascript 中非常常用的语句，表示输出文本。

我们还可以将这个例子写得更加复杂写，不但输出文本，而且输出带 HTML 格式的文本。代码如下：

```
<script type="text/javascript">
   document.write("<h1>Hello, World!</h1>");
</script>
```

演示示例

```
<html>
<head><title>用 document.write 输出带格式的 HTML 文本的 Javascript 示例</title></head>
<body>

<script type="text/javascript">
document.write("<h1>Hello World!</h1>")
</script>

</body>
</html>
```

在参考别人的 Javascript 代码时，你也许会看到<script>里写的不是 type="text/javascript"，而是 language="javascript"。目前这两种方法都可以表示<script></script>里的代码是 Javascript。其中 language 这个属性在 W3C 的 HTML 标准中，已不再推荐使用。

Javascript 写在哪里

Javascript 程序可以放在：

- HTML 网页的<body></body>里
- HTML 网页的<head></head>里
- 外部.js 文件里

Javascript 在<body></body>之间

当浏览器载入网页 Body 部分的时候，就执行其中的 Javascript 语句，执行之后输出的内容就显示在网页中。

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

演示示例

```
<html>
<head><title>用 document.write 输出带格式的 HTML 文本的 Javascript 示例</title></head>
<body>

<script type="text/javascript">
document.write("<h1>Hello World!</h1>")
</script>

</body>
</html>
```

Javascript 在<head></head>之间

有时候并不需要一载入 HTML 就运行 Javascript，而是用户点击了 HTML 中的某个对象，触发了一个事件，才需要调用 Javascript。这时候，通常将这样的 Javascript 放在 HTML 的<head></head>里。

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

```
</script>
</head>
<body>
</body>
</html>
```

演示示例

```
<html>
<head>
<style>
div {border:1px solid #00FF00;width:100px;text-align:center;cursor:hand;}
</style>
<script type="text/javascript">
function clickme()
{
alert("You clicked me!")
}
</script>
</head>

<body>
<p>请点击下面的“click me”。</p>
<div onclick = "clickme()" >click me</div>
</body>
</html>
```

Javascript 放在外部文件里

假使某个 Javascript 的程序被多个 HTML 网页使用，最好的方法，是将这个 Javascript 程序放到一个后缀名为.js 的文本文件里。

这样做，可以提高 Javascript 的复用性，减少代码维护的负担，不必将相同的 Javascript 代码拷贝到多个 HTML 网页里，将来一旦程序有所修改，也只要修改.js 文件就可以，不用再修改每个用到这个 Javascript 程序的 HTML 文件。

在 HTML 里引用外部文件里的 Javascript，应在 Head 里写一句<script src="文件名"></script>，其中 src 的值，就是 Javascript 所在文件的文件路径。示例代码如下：

```
<html>
<head>
<script src="http://book.chinaz.com/html/asdocs/js_tutorials/common.js"></script>
</head>
<body>
</body>
</html>
```

演示示例

```
<html>
<head>
<style>
div {border:1px solid #00FF00;width:100px;text-align:center;cursor:hand;}
</style>
<script src="/html/asdocs/js_tutorials/common.js"></script>
</head>
<body>
<p>请点击下面的“click me”。</p>
<div onclick = "clickme()" >click me</div>
</body>
</html>
```

演示示例里的 common.js 其实就是一个文本文件，内容如下：

```
function clickme()
{
alert("You clicked me!")
}
```

编写 Javascript 代码

象很多其它编程语言一样，Javascript 也是用文本格式编写，由语句 (statements)，语句块 (blocks) 和注释 (comments) 构成。语句块 (blocks) 是由一些相互有关联的语句构成的语句集合。在一句语句 (statement) 里，你可以使用变量，字符串和数字 (literals)，以及表达式 (expressions)。

语句 (Statements)

一个 Javascript 程序就是一个语句的集合。一句 Javascript 语句相当于一句完整的句子。Javascript 语句将表达式 (expressions)用某种方式组合起来，得以完成某项任务。

一句语句 (statement) 包含一个或多个表达式 (expressions)，关键词 (keywords) 和运算符 (operators)。一般来说，一句语句的所有内容写在同一行内。不过，一句语句也可以写成多行。此外，多句语句也可以通过用分号 (;) 分隔，写在同一行内。

建议：将每句语句以显示的方式结束，即在每个语句最后加分号 (;) 来表示该句语句的结束。

以下是几个语句的例子：

```
aBird = "Robin";
```

上面这句语句表示将 "Robin" 这个字符串赋值给变量 aBird。

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

上面这句语句表示将今天的日期值赋值给变量 today。

语句块 (Blocks)

通常来说，用 {} 括起来的一组 Javascript 语句称为语句块 (blocks)。语句块通常可以看做是一句单独的语句。也就是说，在很多地方，语句块可以作为一句单个的语句被其它 Javascript 代码调用。但是以 for 和 while 开头的循环语句例外。另外要注意的是，

注意：在语句块里面的每句语句以分号 (;) 表示结束，但是语句块本身不用分号。

语句块 (blocks) 通常用于函数和条件语句中。

下面的例句中，{} 中间的 5 句语句构成一个语句块 (block)，而最后三行语句，不在语句块内。

```
function convert(inches) {
   feet = inches / 12;
   miles = feet / 5280;
   nauticalMiles = feet / 6080;
   cm = inches * 2.54;
   meters = inches / 39.37;
}
km = meters / 1000;
kradius = km;
```

```
mradius = miles;
```

注释 (Comments)

为了程序的可读性，以及便于日后代码修改和维护时，更快理解代码，你可以在 Javascript 程序里为代码写注释(comments)。

在 Javascript 语言里，用两个斜杠 // 来表示单行注释。见例句：

```
aGoodIdea = "Comment your code thoroughly."; // 这是单行注释。
```

多行注释则用 /* 表示开始， */ 表示结束。见例句：

```
/*
这是多行注释 行一。
这是多行注释 行二。
*/
```

推荐使用多行的单行注释来替代多行注释，这样有助于将代码和注释区分开来。

表达式 (Expressions)

Javascript 表达式 (expressions) 相当于 javascript 语言中的一个短语，这个短语可以判断或者产生一个值，这个值可以是任何一种合法的 Javascript 类型 - 数字，字符串，对象等。最简单的表达式是字符。

表达式示例：

```
3.9                    // 数字字符
"Hello!"                // 字符串字符
false                  // 布尔字符
null                   // null 值字符
{x:1, y:2}              // 对象字符
[1,2,3]                // 数组字符
function(x){return x*x;}  // 函数字符
```

以下是比较复杂的表达式示例：

```
var anExpression = 3 * (4 / 5) + 6;
var aSecondExpression = Math.PI * radius * radius;
var aThirdExpression = aSecondExpression + "%" + anExpression;
var aFourthExpression = "(" + aSecondExpression + ") % (" + anExpression + ")";
```

赋值和等于 (Assignments and Equality)

Javascript 语言中使用等号 (=) 表示变量赋值。等号左边的值可以是：

- 变量
- 数组元素
- 对象属性

等号右边的值可以是任何类型的值，包括表达式。例句如下，表示将整数 8 赋值给 x 这个变量。

```
x = 8;
```

注意： 在 Javascript 里，要判断两个值是否相等，不用等号，而是用两个等号来表示 (==)。 例句如下，表示 x 等于 8。

```
x == 8
```

Javascript 变量(Javascript Variables)

什么是变量?

变量是用来临时存储数值的容器。在程序中，变量存储的数值是可以变化的。

变量的声明(Declaring Variables)

在使用一个变量之前，首先要声明这个变量。Javascript 里，使用 var 来声明变量。

声明变量有以下几种方法：

1. 一次声明一个变量。例句如下：

```
var a;
```

2. 同时声明多个变量，变量之间用逗号相隔。例句如下：

```
var a, b, c;
```

3.声明一个变量时，同时赋予变量初始值。例句如下：

```
var a=2;
```

4. 同时声明多个变量，并且赋予这些变量初始值，变量之间用逗号相隔。例句如下：

```
var a=2, b=5;
```

变量的命名规则

变量名可以是任意长度。变量名必须符合下列规则：

- 变量名的第一个字符必须是英文字母，或者是下划线符号(underscore)_
- 变量名的第一个字母不能是数字。其后的字符，可以是英文字母，数字，和下划线符号符号(underscore)_
- 变量名不能是 Javascript 的保留字(参见 Javascript 保留字)。

注意：Javascript 代码是区分大小写的(case-sensitive)。变量 myname 和 MyName 表示的是两个不同的变量。写错变量的大小写，是初学者最常见的错误之一。

Javascript 常用运算符(Operators)

算术运算符

运算符	运算符说明	示例	示例说明
+	加法	x+y	如果 x 为整数2,y 为整数5, x+y 等于7 如果 x 为字符串"text1", y 为字符串"fun", x+y 则等于"text1fun"
-	减法	x-y	
*	乘法	x*y	
/	除法	x/y	
%	两者相除求余数	x%y	如果 x 等于10, y 等于3, x%y 结果等于1
++	递增	x++	如果 x 等于10, x++等于11
--	递减	y--	如果 y 等于10, y--等于9

逻辑运算符

运算符	运算符说明	示例	示例说明
==	等于	x==y	如果 x 等于2, y 等于2,则 x==y
===	全等于(值相等,数据类型也相等)	x===y	如果 x 等于整数2,y 为字符串"2", 则 x===y 不成立
>	大于	x>y	
>=	大于等于	x>=y	
<	小于	x<y	
<=	小于等于	x<=y	
!=	不等于	x!=y	
!==	不全等于	x!==y	
&&	与(and)	x < 10 && y > 1	
!	非(not)	!(x==y)	
||	或(or)	x==8 || y==8	

赋值运算符

运算符	运算符说明	示例	示例说明
=	赋值	x=5	将整数5这个值赋给变量 x

注意：请注意赋值(=)和等于(==)的区别。

Javascript 条件语句(Javascript Conditional Statements)

在一般情况下，程序语句的执行是按照其书写顺序来执行的。前面的代码先执行，后面的代码后执行。但是这种简单的自上而下的单向流程只适于用一些很简单的程序。大多数情况下，需要根据逻辑判断来决定程序代码执行的优先顺序。要改变程序代码执行的先后顺序，任何编程语言都需要用到条件语句和循环语句，Javascript 也不例外。

这一节我们主要介绍 Javascript 条件语句。

Javascript 条件语句有以下几种：

- 单项条件结构 (if 条件语句)
- 双向条件结构 (if...else 条件语句)
- 多项条件结构 (switch 条件语句)

单项条件结构 (if 条件语句)

If 条件语句的语法如下：

```
if (expression)
 {
  statement1
 }
```

这句语法的含义是，如果符合 expression 条件，就执行 statement1代码，反之，则不执行 statement1代码。

下面的这个 Javascript 示例就用到了 Javascript 的 if 条件语句。首先用.length 计算出字符串 What's up?的长度，然后使用 if 语句进行判断，如果该字符串长度<100，就显示"该字符串长度小于100。"。

```
<html>
<head><title>一个使用到 if 条件语句的 Javascript 示例</title></head>
<body>
<script type="text/javascript">
var vText = "What's up?";
var vLen = vText.length;
if (vLen < 100)
{
document.write("<p> 该字符串长度小于100。</p>")
}
</script>
</body>
</html>
```

演示示例

```
<html>
<head>
<title>一个使用到 if 条件语句的 Javascript 示例</title>
</head>
<body>

<script type="text/javascript">
var vText = "What's up?";
var vLen = vText.length;
if (vLen < 100)
{
document.write("<p>  该字符串长度小于 100。</p>")
}
</script>

<p>Javascript 示例代码解释：这个 Javascript 示例用到了 Javascript 的 if 条件语句。
首先用.length 计算出字符串 What's up?的长度，然后使用 if 语句。if 语句的内容是：如果该字符串长度<100，就显示"该字符串长度小于 100。"。
</p>

</body>
</html>
```

双向条件结构 (if...else 条件语句)

If...else 条件语句的语法如下：

```
if (expression)
  {
   statement1
  }
else
  {
   statement2
  }
```

这句语法的含义是，如果符合 expression 条件，则执行 statement1代码，反之，则执行 statement2 代码。

下面的 Javascript 示例使用了 if...else 条件语句判断，如果 vHour 小于17，显示"日安"，反之则显示"晚安"。

```
<html>
<head><title>使用 if...else 条件语句的 Javascript 示例</title></head>
```

```
<body>
<script type="text/javascript">
var vDay = new Date()
var vHour = vDay.getHours()
if (vHour < 17)
{
document.write("<b>日安</b>")
}
else
{
document.write("<b>晚安</b>")
}
</script>
</body>
</html>
```

演示示例

```
<html>
<head><title>使用 if...else 条件语句的 Javascript 示例</title></head>
<body>

<script type="text/javascript">
var vDay = new Date()
var vHour = vDay.getHours()

if (vHour < 17)
{
document.write("<b>日安</b>")
}
else
{
document.write("<b>晚安</b>")
}
</script>

<p>Javascript 代码示例解释：首先将今天的日期赋值给变量 vDay，然后用.getHours 得出 vDay 的小时数，赋值给变量 vHour，然后使用 if...else 条件语句判断，如果 vHour 小于 17，显示"日安"，反之则显示"晚安"。</p>

</body>
</html>
```

多项条件结构 (switch 条件语句)

Switch 条件语句的语法如下：

```
switch (expression)
 {
   case label1 :
     statement1
     break
   case label2 :
     statement2
     break
   ...
   default :
     statementdefault
 }
```

这句语法的含义是，如果 expression 等于 label1，则执行 statement1代码；如果 expression 等于 label2，则执行 statement2代码；以此类推。如果 expression 不符合任何 label，则执行 default 内的 statementdefault 代码。 Switch 条件语句中的 break，表示 switch 语句结束。如果没有使用一个 break 语句，则多个 label 块被执行。

下面的 Javascript 示例使用了 switch 条件语句，根据星期天数的不同，显示不同的话。

```
<html>
<head><title>使用 swith 条件语句的 Javascript 示例</title></head>
<body>
<script type="text/javascript">
var d = new Date()
theDay=d.getDay()
switch (theDay)
{
case 5:
document.write("<b>总算熬到星期五了。</b>")
break
case 6:
document.write("<b>哈哈，周末啦！</b>")
break
case 0:
document.write("<b>明天又要上班，想想就烦。</b>")
break
default:
document.write("<b>每个工作日慢得都象蜗牛爬啊！</b>")
}
</script>
</body>
</html>
```

演示示例

```
<html>
<head><title>使用 swith 条件语句的 Javascript 示例</title></head>
<body>
<script type="text/javascript">
var d = new Date()
theDay=d.getDay()
switch (theDay)
{
case 5:
document.write("<b>总算熬到星期五了。</b>")
break
case 6:
document.write("<b>哈哈，周末啦！</b>")
break
case 0:
document.write("<b>明天又要上班，想想就烦。</b>")
break
default:
document.write("<b>每个工作日慢得都象蜗牛爬啊！</b>")
}
</script>
<p>该 Javascript 示例解释：首先将今天的日期值赋值给变量 d，然后用.getDay 得出天数，赋值给变量 theDay，然后使用 switch 条件语句。如果 theDay 等于 5，表示是星期五；如果是 6，表示是星期六；如果是 0，表示是星期天；如果是其它数，表示是星期一到星期四。根据值的不同，显示不同的内容。</p>
</body>
</html>
```

Javascript 循环语句 (Javascript Loop Statements)

在一般情况下，程序语句的执行是按照其书写顺序来执行的。前面的代码先执行，后面的代码后执行。但是这种简单的自上而下的单向流程只适于用一些很简单的程序。大多数情况下，需要根据逻辑判断来决定程序代码执行的优先顺序。要改变程序代码执行的先后顺序，任何编程语言都需要用到条件语句和循环语句，Javascript 也不例外。

这一节我们主要介绍 Javascript 循环语句。

Javascript 循环语句有以下几种：

- 在循环的开头测试表达式 (while 循环语句)
- 在循环的末尾测试表达式 (do...while 循环语句)
- 对对象的每个属性都进行操作 (for...in 循环语句)
- 由计数器控制的循环 (for 循环语句)

使用 for 循环语句

for 循环语句指定了一个计数器变量，一个测试条件，和更新计数器的行为。

每次循环重复之前，都要测试条件。如果测试成功，则执行循环内的代码；如果测试不成功，则不执行循环内的代码，而是执行紧跟在循环后的第一行代码。当执行该循环时，计数器变量在下次重复循环前被更新。

如果循环条件一直不满足，则永不执行该循环。如果条件一直满足，则会导致无限循环。前一种，在某种情况下是需要的，但是后一种，基本不应发生，所以写循环条件时一定要注意。

for 循环语句示例代码：

```
<html>
<head><title>一个使用到 for 循环的 Javascript 示例</title></head>
<body>
<p>
<script type="text/javascript">
for (i = 0; i <= 5; i++)
{
document.write(i)
document.write("<br>")
}
</script>
</p>
</body>
</html>
```

演示 for 循环语句示例

```
<html>
<head>
<title>一个使用到 for 循环的 Javascript 示例</title>
</head>
<body>
<p>
<script type="text/javascript">
for (i = 0; i <= 5; i++)
{
document.write(i)
document.write("<br>")
}
</script>
</p>

<p>Javascript 示例代码解释：这个 Javascript 示例用到了 for 循环语句。
循环语句允许重复执行一行或数行代码，for 循环要使用一个计数器变量，每重复一次循环之后，计数器变量的值就会增加或者减少。这个 Javascript 示例中，计数器变量为 i，i 初始值为 0，i++表示每次重复执行后 i 的值就加 1，终止循环条件为 i<=5，也就是说，一旦 i 的值大于 5，就终止循环。该示例中，重复循环的语句是 for 循环里面的两句 document.write 语句。</p>

</body>
</html>
```

使用 for...in 循环语句

Javascript 提供了一种特别的循环方式来遍历一个对象的所有用户定义的属性或者一个数组的所有元素。for...in 循环中的循环计数器是一个字符串，而不是数字。它包含了当前属性的名称或者表示当前数组元素的下标。

for...in 循环语句示例代码：

```
<html>
<head><title>一个使用到 for...in 循环的 Javascript 示例</title></head>
<body>
<script type="text/javascript">
// 创建一个对象 myObject 以及三个属性 sitename, siteurl, sitecontent。
var myObject = new Object();
myObject.sitename = "站长站 站长书库";
myObject.siteurl = "book.chinaz.com";
myObject.sitecontent = "网页教程代码图库的中文站点";
//遍历对象的所有属性
for (prop in myObject)
{
```

```
    document.write("属性 '" + prop + "' 为 " + myObject[prop]);
    document.write("<br>");
}
</script>
</body>
</html>
```

演示 for...in 循环语句示例

```
<html>
<head>
<title>一个使用到 for...in 循环的 Javascript 示例</title>
</head>
<body>
<script type="text/javascript">
// 创建一个对象 myObject 以及三个属性 sitename, siteurl, sitecontent。
var myObject = new Object();
myObject.sitename = "站长站 站长书库";
myObject.siteurl = "book.chinaz.com";
myObject.sitecontent = "网页教程代码图库的中文站点";
//遍历对象的所有属性
for (prop in myObject)
{
document.write("属性 '" + prop + "' 为 " + myObject[prop]);
document.write("<br>");
}
</script>
</body>
</html>
```

使用 while 和 do...while 循环语句

while 循环和 for 循环类似。其不同之处在于， while 循环没有内置的计数器或更新表达式。如果你希望控制语句或语句块的循环执行，不只是通过 “运行该代码 n 次”这样简单的规则，而是需要更复杂的规则，则应该用 while 循环。

注意：由于 while 循环没有显式的内置计数器变量，因此比其它类型的循环更容易产生无限循环。此外，由于不易发现循环条件是在何时何地被更新的，很容易编写一个实际上从不更新条件的 while 循环。因此在编写 while 循环时应特别小心。

while 循环语句示例代码：

```
<html>
<head><title>一个使用到 while 循环的 Javascript 示例</title></head>
<body>
<p>
<script type="text/javascript">
```

```
i = 0
while (i <= 5)
{
document.write(i + "<br>")
i++
}
</script>
</p>
</body>
</html>
```

演示 while 循环语句示例

```
<html>
<head>
<title>一个使用到 while 循环的 Javascript 示例</title>
</head>
<body>
<p>
<script type="text/javascript">
i = 0
while (i <= 5)
{
document.write(i + "<br>")
i++
}
</script>
</p>
<p>Javascript 示例代码解释：这个 Javascript 示例用到了 while 循环语句。
循环语句允许重复执行一行或数行代码，while 后面紧跟的是终止循环的条件。这个 Javascript 示例中，设一个变量为 i，i 初始值为 0，i++表示每次重复执行后 i 的值就加 1，终止循环条件为 while (i <= 5)，也就是说，一旦 i 的值大于 5，就终止循环。该示例中，重复循环的语句是 while 循环里面的 document.write 语句。</p>
</body>
</html>
```

在 JScript 中还有 do...while 循环与 while 循环相似，不同处在于它总是至少运行一次，因为是在循环的末尾检查条件，而不是在开头。

do...while 循环语句示例：

```
<html>
<head><title>一个使用到 do...while 循环的 Javascript 示例</title></head>
<body>
<p>
<script type="text/javascript">
```

```
i = 0
do
{
document.write( i + "<br>")
i++
}
while (i <= 5)
</script>
</body>
</html>
```

演示 do...while 循环语句示例

```
<html>
<head>
<title>一个使用到 do...while 循环的 Javascript 示例</title>
</head>
<body>
<p>
<script type="text/javascript">
i = 0
do
{
document.write( i + "<br>")
i++
}
while (i <= 5)
</script>
</p>
<p>Javascript 示例代码解释：这个 Javascript 示例用到了 do...while 循环语句。
循环语句允许重复执行一行或数行代码，do 后面跟的是重复执行的代码，while 后面跟的是终止循环的条件。这个 Javascript 示例中，设一个变量为 i，i 初始值为 0，i++表示每次重复执行后 i 的值就加 1，终止循环条件为 while (i <= 5)，也就是说，一旦 i 的值大于 5，就终止循环。该示例中，重复循环的语句是 while 循环里面的 document.write 语句。</p>
</body>
</html>
```

使用 break 和 continue 语句

在 Javascript 中，当某些条件得到满足时，用 break 语句来中断一个循环的运行。（请注意，也用 break 语句退出一个 switch 块。参见 Javascript 条件语句）。如果是一个 for 或者 for...in 循环，在更新计数器变量时使用 continue 语句越过余下的代码块而直接跳到循环的下一次重复中。

break 示例代码：

```
<html>
```

```
<head><title>一个用 break 中断循环的 Javascript 示例代码</title>
<script type="text/javascript">
function BreakTest(breakpoint){
   var i = 0;
   var m = 0;
   while (i < 100)
   {
   //当 i 等于 breakpoint 时，中断循环
   if (i == breakpoint)
      break;
      m=m + i;
      i++;
   }
   return(m);
}
</script>
</head>
<body>

<script  type="text/javascript">
//设函数 BreakTest 参数 breakpoint 值为23，得到从1加到22的合计。
document.write(BreakTest(23))
</script>

</body>
</html>
```

演示 break 示例

```
<html>
<head>
<title>一个用 break 中断循环的 Javascript 代码示例</title>
<script type="text/javascript">
function BreakTest(breakpoint){
var i = 0;
var m = 0;
while (i < 100)
{
//当 i 等于 breakpoint 时，中断循环
if (i == breakpoint)
break;
m=m + i;
i++;
}
return(m);
}
</script>
</head>
<body>

<script  type="text/javascript">
//设函数 BreakTest 参数 breakpoint 值为 23，得到从 1 加到 22 的合计。
document.write(BreakTest(23))
</script>

</body>
</html>
```

continue 示例代码：

```
<html>
<head>
<title>一个用 continue 跳过后面代码，开始循环的下一次重复的 Javascript 代码</title>
</head>
<body>

<script type="text/javascript">
//该 script 代码用来输出1到10之间的奇数。
var x ;
for (x=1; x<10; x++)
{
//如果 x 被2整除，则跳过后面代码，开始下一次重复；
//如果 x 不能被2整除，则执行后面代码，输出 x 。
```

```
if (x%2==0)
   continue;
   document.write (x + "<br>");
}
</script>

</body>
</html>
```

演示 continue 示例

```
<html>
<head>
<title>一个用 continue 跳过后面代码，开始循环的下一次重复的 Javascript 示例代码</title>
</head>
<body>

<script type="text/javascript">
//这段 Javascript 代码用来输出 1 到 10 之间的奇数。
var x ;
for (x=1; x<10; x++)
{
//如果 x 被 2 整除，则跳过后面代码，开始下一次重复；
//如果 x 不能被 2 整除，则执行后面代码，输出 x 。
if (x%2==0)
continue;
document.write (x + "<br>");
}
</script>

</body>
</html>
```

Javascript 保留字(Javascript Reserved Words)

Javascript 保留字(Javascript Reserved Words)

Javascript 保留字(Reserved Words)是指在 Javascript 语言中有特定含义，成为 Javascript 语法中一部分的那些字。Javascript 保留字是不能作为变量名和函数名使用的。使用 Javascript 保留字作为变量名或函数名，会使 Javascript 在载入过程中出现编译错误。

Javascript 保留字列表：

break	delete	function	return	typeof
case	do	if	switch	var
catch	else	in	this	void
continue	false	instanceof	throw	while
debugger	finally	new	true	with
default	for	null	try	

Javascript 未来保留字(Javascript Future Reserved Words)

Javascript 还有一些未来保留字，这些字虽然现在没有用到 Javascript 语言中，但是将来有可能用到。

Javascript 未来保留字列表：

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