第 5 章　JavaScript 的流程控制语句

因为 JavaScript 是一种完整的语言，所以它也可以控制程序的流程，实现分支选择和循环判断。分支选择就是有多条路可以走，但是程序只能走一条路，如今天如果下雨就在家看电影，如果不下雨就出去逛街，是否下雨是一个判断条件，看电影和逛街是两条“路”，但是只能走一条路。循环判断语句是根据一个条件来判断，是否重复执行某一条路。

本章主要涉及到的知识点有：

- 掌握判断语句的执行顺序
- 学习 if 判断语句
- 学习 switch 多条件判断语句
- 掌握循环语句的执行顺序
- 学习 while 循环语句
- 学习 for 循环语句

5.1　分 支 语 句

在编写一个程序时，通常需要根据特定的条件执行不同的语句，或者一段语句。比如性别为“男”时，称呼用户为“先生”；性别为“女”时，称呼用户为“女士”。前面的章节里也有很多类似的例子，比如在第 2 章里一个页面上，在性别上选择“男”或者“女”时，页面本身将会根据实际情况显示，或者隐藏某些页面元素，同时在计算时的算法也有所不同。摘取其中的一段代码如下所示：

```
01  //根据不同情况作相应计算
02  if( sex1 == true ){
03      //选择了“男”时
04      result = yourmoney - yoursmoke - yourwine;
05  }
06  if( sex2 == true ){
07      //选择了“女”时
08      result = yourmoney - yourface - yourclothe;
09  }
```

可以看到上面使用了 if 语句来进行条件判断，但是分支语句不仅仅只有这些，本节将会依次介绍。

5.1.1　使用 if 实现条件判断

在条件语句中，if 语句是使用得最广泛也是比较简单的一个语句了，if 语句从英文的

字面意思来看，其意义也很明确，就是“如果”，也可以这样来理解程序，看下面的代码：

```
if( i > 2 ){
    a = i * 10;
}
```

上面的代码的意义是，如果变量 i 大于 2，那么变量 a 的值就是 i 与 10 的乘积。可见，if 语句是很好理解的。if 语句的语法如下所示：

```
if( 条件表达式 ){
    语句或语句块;
}
```

从上面的语法可以看出来，if 语句包含 3 个部分：if 关键字、包含在圆括号里的条件表达式，以及包含在大括号里的语句或语句块，这些语句或者语句块是在条件表达式为 true 时需要执行的。如果条件表达式的值为 false，那么大括弧里的语句或语句块是不会被执行的。同时，在 if 语句前后可能还有其他的语句，前后的语句都不会受 if 语句里条件表达式的影响，即使 if 语句设定的需要执行的语句，或者语句块因为条件表达式为 false，导致无法执行，if 语句后的其他语句仍然会继续执行。看下面的代码段：

```
01  <script language="JavaScript">
02  <!--
03  var i = 3;
04  var a = 10;
05  if( i > 2 ){
06      a = i * 10;
07  }
08  alert(a);                    //输出 a 的值
09  //-->
10  </script>
```

在第 8 行有一条“alert(a);”语句，在 if 语句之后。通过运行代码可以看到，if 语句执行后由于条件表达式为 true，因此 a 的值被改变了，结果如图 5.1 所示。

图 5.1　if 语句示例

在 JavaScript 里，如果条件里需要执行的语句只有一句，则可以省略外面的大括弧，上面的条件部分代码同样可以写成如下的样子：

```
f( i > 2 )
    a = i * 10;
```

按照上面的规律，可以循环嵌套，也就是说，如果 if 语句里面的代码又是 if 语句，仍然可以使用同样的规则，看下面的代码：

```
if( i > 2 )
    if( j < 3 )
        a = i * 10;
```

上面也是正确的写法，但是，为了使程序可读性更强，最好还是不要省略大括弧，推荐的写法如下所示：

```
if( i > 2 ){
    if( j < 3 ){
        a = i * 10;
    }
}
```

在条件判断里，通常在 if 的条件表达式里不仅仅使用一个表达式，还可以用多个子表达式联合起来进行条件判断，这需要使用逻辑运算符，看下面的代码：

```
<script language="JavaScript">
<!--
var i = 3;
var j = 10;
var a = 0;
if( i < 4 && j > 5 ){
    a = i * j;
}
//-->
</script>
```

上面的代码段第 6 行使用逻辑与运算符“&&”，将两个子表达式“i < 4”和“j > 5”连接为一个整体。只有当两个子表达式都为 true 时，整个表达式的值才为 true。当需要判断的条件较多时，可以并列使用多个 if 语句来进行控制。下面通过一个实例来结束本小节的内容。

【范例 5-1】 这个实例是一个在线的小测试程序。HTML 文件见 5-1.html，代码如下所示：

```
<html>
<head>
<title>if 条件语句示例</title>
<script language="JavaScript">
<!--
//地点检查
function checkCity(v){
    if( v == "a" ){
        alert("回答正确! ");
    }
    if( v == "b" ){
        alert("回答错误! ");
    }
    if( v == "c" ){
        alert("回答错误! ");
    }
    if( v == "d" ){
        alert("回答错误! ");
    }
}
//时间检查
function checkDay(v){
    if( v == "a" ){
        alert("回答错误! ");
    }
    if( v == "b" ){
```

```
27          alert("回答错误！");
28      }
29      if( v == "c" ){
30          alert("回答正确！");
31      }
32      if( v == "d" ){
33          alert("回答错误！");
34      }
35  }
36  //-->
37  </script>
38  </head>
39  <body>
40  <h1>测试题</h1>
41  <hr>
42  1、2016 年奥运会在哪个城市举行？<br>
43  <input name="city" type="radio" value="a" onclick="checkCity(this.
    value)"> 里约<br>
44  <input name="city" type="radio" value="b" onclick="checkCity(this.
    value)">悉尼<br>
45  <input name="city" type="radio" value="c" onclick="checkCity(this.
    value)">纽约<br>
46  <input name="city" type="radio" value="d" onclick="checkCity(this.
    value)">伦敦<br>
47
48  2、2008 年奥运会什么时候开幕的？<br>
49  <input name="day" type="radio" value="a" onclick="checkDay(this.
    value)">7 月 1 号<br>
50  <input name="day" type="radio" value="b" onclick="checkDay(this.
    value)">8 月 1 号<br>
51  <input name="day" type="radio" value="c" onclick="checkDay(this.
    value)">8 月 8 号<br>
52  <input name="day" type="radio" value="d" onclick="checkDay(this.
    value)">9 月 1 号<br>
53  </body>
54  </html>
```

本例子在页面上放置了两个问题，分别是针对 2016 年奥运会的举办城市和 2008 年奥运会的开幕时间。每个问题设置了 4 个选项，其中只有一个才是正确的。用户单击选项，网页则会判断用户的选择，提示用户的选择正确与否。

为了根据用户的选择进行判断，还另外编写了两个函数来进行判断，分别是第 7～20 行的 checkCity 和第 22～35 行的 checkDay。在函数体内使用了并列的 if 语句来对各个选项进行判断，为了触发函数，使用了单选按钮的 click 事件，两个函数都有一个参数，用来传递用户当前选择的选项对应的答案。因此，在给每个单选选项添加 click 事件与函数进行关联时，还使用了 this.value 来获取当前选项的值。运行 5-1.html 单击选项的情况，如图 5.2 所示。

感兴趣的读者，可以模仿 5-1.html，设置自己的测试题目和答案，加深体会。

5.1.2　使用 if…else 实现两个分支条件

if 语句的后面，还可以跟一个 else 分句，当 if 语句所包含的条件表达式为 false 时，用

来执行 else 分句包含的语句或语句块。也就是说使用 if...else 语句可以根据条件表达式的值为 true 或者 false 分别执行相应的语句或语句块。下面是 if...else 语句的语法结构：

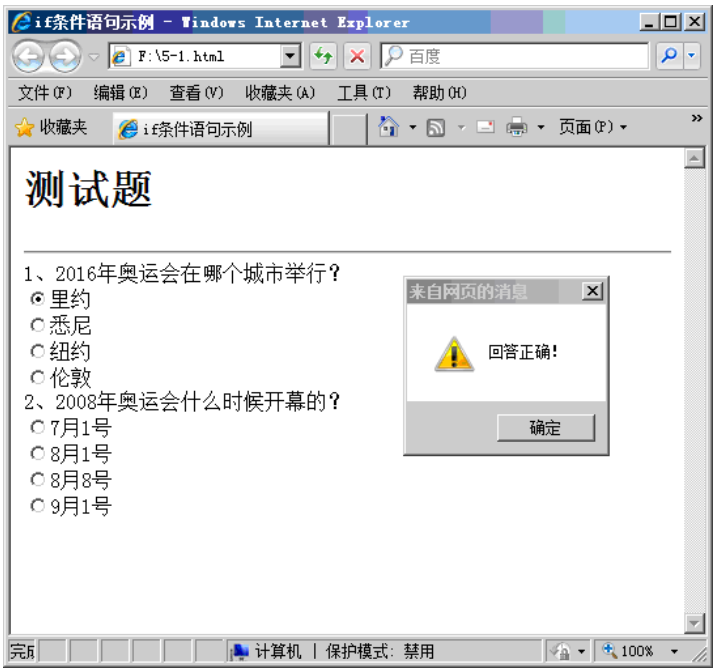

图 5.2　if 语句测试题示例

```
if( 条件表达式 ){
    条件表达式为 true 时语句或语句块;
}else{
    条件表达式为 false 时语句或语句块;
}
```

if...else 语句也很简单，下面用示例进行说明。

1．条件表达式为true

当条件表达式满足指定的规则时，执行第一个分句的内容。

```
01  <script language="JavaScript">
02  <!--
03  //条件表达式为 true
04  var i = 3;
05  if( i > 2 ){
06      alert("变量 i 大于 2。");                    //输出提示
07  }else{
08      alert("变量 i 小于或等于 2。");              //输出提示
09  }
10  //-->
11  </script>
```

上面的代码第 4 行定义了变量 i，赋值为 3。第 5 行的条件表达式为“i＞2”，这个条件表达式的明显为 true，因此执行第 6 行语句，而 else 分句里的第 8 行则得不到执行。

2．条件表达式为false

当条件表达式不满足指定的规则时，执行第二个分句，即 else 分句的内容。

```
<script language="JavaScript">
<!--
//条件表达式为 false
var j = 3;
if( j < 2 ){
    alert("变量 j 小于 2。");                    //输出提示
}else{
    alert("变量 j 大于或等于 2。");              //输出提示
}
//-->
</script>
```

上面代码第 4 行定义了一个变量 j，并赋值为 3。第 5 行条件表达式为“j < 2”，这个条件表达式的值为 false，因此第一个大括弧里第 6 行语句得不到执行，而 else 分句大括弧里的第 8 行语句得到执行。

5.1.3　if 和 if…else 的嵌套

在使用 if 或者 if...else 等语句进行条件判断时，在较为复杂一点的时候，可能会需要进行进一步的判断。比如小朋友们做游戏需要分组，小朋友们的年龄是 3~10 岁。假设分组的规则是 5 岁以下分为第一组；对于 5 岁以上的小朋友，如果小于 8 岁，那么分为第二组；大于 8 岁，分为第三组。对于这样一个逻辑，就可以使用嵌套来编写，下面结合汉字描述程序的逻辑结构：

```
if( 年龄 > 5 ){
    if( 年龄 > 8 ){
        分到第三组;
}else{
        分到第二组;
}
}else{
    分到第一组;
}
```

可以看到，上面的代码段，在第一个 if 关键字所在的大括弧内，嵌套了一个 if...else 语句。通过这样的嵌套，对条件进行了进一步的判断。嵌套是不限制层级的，可以无限次地进行嵌套，见下面的代码段：

```
<script language="JavaScript">
<!--
var city = "华盛顿";
if( city == "北京" ){
    alert("中国首都！");                    //输出提示
}else{
    if( city == "东京" ){
        alert("日本首都！");                //输出提示
    }else{
```

```
10          if( city == "多伦多" ){
11              alert("加拿大首都！");
12          }else{
13              if( city == "华盛顿" ){
14                  alert("美国首都！");
15              }else{
16                  alert("未知城市！");
17              }
18          }
19      }
20  }
21  //-->
22  </script>
```

上面的代码第 4~20 行使用了三层 if...else 语句嵌套，用来判断所给出的城市对应的首都。可以看到，这样的代码有些眼花缭乱，清晰度不够高，而且嵌套的时候如果不注意代码缩进，很容易造成混淆，上面的代码如果不考虑缩进后情况如下所示：

```
<script language="JavaScript">
<!--
var city = "华盛顿";
if( city == "北京" ){
alert("中国首都！");
}else{
if( city == "东京" ){
alert("日本首都！");
}else{
if( city == "多伦多" ){
alert("加拿大首都！");
}else{
if( city == "华盛顿" ){
alert("美国首都！");
}else{
alert("未知城市！");
}
}
}
}
//-->
</script>
```

这样执行仍然不会有任何问题，但是阅读起来显得非常费劲，因此有了一种改进的方案，就是把被嵌套的 if 和上一个 else 放在同一行，中间用空格隔开。上面的代码段按照这样的规则改进后如下所示：

```
01  <script language="JavaScript">
02  <!--
03  var city = "华盛顿";
04  if( city == "北京" ){
05      alert("中国首都！");
06  }else if( city == "东京" ){
07      alert("日本首都！");
08  }else if( city == "多伦多" ){
09      alert("加拿大首都！");
10  }else if( city == "华盛顿" ){
```

```
11      alert("美国首都！");
12  }else{
13      alert("未知城市！");
14  }
15  //-->
16  </script>
```

可以看到，经过改进后的代码段，不仅仅是篇幅缩短，而且代码本身也变得清晰易读，有点类似 if 语句并列的情况。

【范例 5-2】 考虑到这种写法的好处，现在将 5.1.1 中的例子 5-1.html 改进一下，保存为 5-2.html，代码如下所示：

```
01  <html>
02  <head>
03  <title>条件语句嵌套示例</title>
04  <script language="JavaScript">
05  <!--
06  //地点检查
07  function checkCity(v){
08      if( v == "a" ){
09          alert("回答正确！");
10      }else if( v == "b" ){
01          alert("回答错误！");
02      }else if( v == "c" ){
03          alert("回答错误！");
04      }else if( v == "d" ){
05          alert("回答错误！");
06      }
07  }
08  //时间检查
09  function checkDay(v){
10      if( v == "a" ){
11          alert("回答错误！");
12      }else if( v == "b" ){
13          alert("回答错误！");
14      }else if( v == "c" ){
15          alert("回答正确！");
16      }else if( v == "d" ){
17          alert("回答错误！");
18      }
19  }
20  //-->
21  </script>
22  </head>
23  <body>
24  <h1>测试题</h1>
25  <hr>
26  1、2016 年奥运会在哪个城市举行？<br>
27  <input name="city" type="radio" value="a" onclick="checkCity(this.
    value)">里约<br>
28  <input name="city" type="radio" value="b" onclick="checkCity(this.
    value)">悉尼<br>
29  <input name="city" type="radio" value="c" onclick="checkCity(this.
    value)">纽约<br>
30  <input name="city" type="radio" value="d" onclick="checkCity(this.
    value)">伦敦<br>
31
```

```
32  2、2008 年奥运会什么时候开幕的？<br>
33  <input name="day" type="radio" value="a" onclick="checkDay(this.
    value)">7 月 1 号<br>
34  <input name="day" type="radio" value="b" onclick="checkDay(this.
    value)">8 月 1 号<br>
35  <input name="day" type="radio" value="c" onclick="checkDay(this.
    value)">8 月 8 号<br>
36  <input name="day" type="radio" value="d" onclick="checkDay(this.
    value)">9 月 1 号<br>
37  </body>
38  </html>
```

感兴趣的读者可以运行后查看效果，检验与 5-1.html 是否一样。

5.1.4　使用 switch 实现多分支判断

在 JavaScript 中，switch 语句是一个经常使用的条件控制语句。switch 跟 if 或者 if...else 语句较大的区别就是，switch 语句是根据一个固定的表达式的值来进行条件控制的，不像 if 或者 if...else 语句那样能够使用并列或者嵌套对多个表达式进行判断。因此，switch 对于只有一个单一表达式的条件控制尤其有用。

比如 5.1.1 小节里的用户选择的值“v”的判断，5.1.3 小节里的对于各国首都判断的例子，就是针对固定的表达式的值来进行判断的，在这些情况下，都可以使用 switch 语句。switch 语句的语法如下所示：

```
switch  ( 表达式 ){
    case 备选值 1:
        语句或语句块;
        break;
    case 备选值 2:
        语句或语句块;
        break;
    ...
case 备选值 n:
        语句或语句块;
        break;
    default:
        默认执行语句或语句块;
}
```

一个完整的 switch 语句包括一个 switch 关键字、一个需要判断的表达式、一个开始大括弧、若干个判断表达式备选值的 case 标签、符合每个 case 标签匹配的值时需要执行的语句或语句块、匹配 case 标签的 break 关键字、一个 default 标签以及一个结束的大括弧。其中 default 标签是用来在当每个 case 标签所匹配的值都不符合表达式的值时，设置默认的执行语句，default 不是必须的，可以省略。

【范例 5-3】 下面使用 switch 对 5.1.1 小节和 5.1.3 小节中的两个实例进行改造，用来理解 switch 语句的功能。对于 5-1.html 进行改进，保存 HTML 文件为 5-3.html，代码如下所示：

```
01  <html>
02  <head>
```

```
<title>switch 条件语句嵌套示例</title>
<script language="JavaScript">
<!--
//地点检查
function checkCity(v){
    switch( v ){
        case "a":
            alert("回答正确! ");
            break;
        case "b":
            alert("回答错误! ");
            break;
        case "c":
            alert("回答错误! ");
            break;
        case "d":
            alert("回答错误! ");
            break;
    }
}
//时间检查
function checkDay(v){
    switch( v ){
        case "a":
            alert("回答错误! ");
            break;
        case "b":
            alert("回答错误! ");
            break;
        case "c":
            alert("回答正确! ");
            break;
        case "d":
            alert("回答错误! ");
            break;
    }
}
//-->
</script>
</head>
<body>
<h1>测试题</h1>
<hr>
1、2016 年奥运会在哪个城市举行? <br>
<input name="city" type="radio" value="a" onclick="checkCity(this.
value)">里约<br>
<input name="city" type="radio" value="b" onclick="checkCity(this.
value)">悉尼<br>
<input name="city" type="radio" value="c" onclick="checkCity(this.
value)">纽约<br>
<input name="city" type="radio" value="d" onclick="checkCity(this.
value)">伦敦<br>

2、2008 年奥运会什么时候开幕的? <br>
<input name="day" type="radio" value="a" onclick="checkDay(this.
value)">7 月 1 号<br>
<input name="day" type="radio" value="b" onclick="checkDay(this.
value)">8 月 1 号<br>
```

```
55  <input name="day" type="radio" value="c" onclick="checkDay(this.
    value)">8 月 8 号<br>
56  <input name="day" type="radio" value="d" onclick="checkDay(this.
    value)">9 月 1 号<br>
57  </body>
58  </html>
```

可以看到，在两个函数 checkCity 和 checkDay 里，都使用了 switch 语句。执行后效果与原来一样。下面对 5.1.3 小节中的例子进行改进，代码如下所示：

```
01  <script language="JavaScript">
02  <!--
03  var city = "华盛顿";
04  switch( city ){
05      case "北京":
06          alert("中国首都！");
07          break;
08      case "东京":
09          alert("日本首都！");
10          break;
11      case "多伦多":
12          alert("加拿大首都！");
13          break;
14      case "华盛顿":
15          alert("美国首都！");
16          break;
17      default:
18          alert("未知城市！");
19  }
20  //-->
21  </script>
```

读者可以运行后看看情况，然后自行编写后调试。

5.2　循 环 语 句

在本节之前所学习到的内容，都是从上到下执行的语句，或者通过条件选择，进行分支上的控制，但是总的规律是从上至下的执行方式。在现实生活中，经常需要重复性、有规律地做一些事情，比如每天都要按时起床、吃早饭、上班或上学等等，周而复始。同样的道理，程序里也经常会出现重复的情况，比如重复让某一个变量乘以一个整数，重复 100 遍等等。本节的内容，就是介绍循环语句来实现重复的动作。

5.2.1　while 循环

while 语句是一个比较简单的循环语句，while 语句的规则是当某个给定的条件表达式为 true 时，重复执行一条语句或者语句块。下面先看一下 while 语句的语法结构：

```
while( 条件表达式 ){
    语句或语句块;
}
```

可以看到，while 语句的语法跟 if 语句类似，都是 while 关键字后面紧跟一个圆括弧包含的条件表达式，当条件表达式为 true 时，执行大括弧里的语句或语句块，不同的是，while 语句会重复执行大括弧里的语句或语句块一直到条件表达式为 false 为止。因此，通常需要设置一个类似计数器的变量来组成条件表达式，从而当达到某个数量以后，停止循环。

【范例 5-4】 下面是一个简单的例子。

```
<script language="JavaScript">
<!--
var i = 1;
var num = 10;
while( i < 5 ){
    i++;                    //执行 i 自增运算
    num = num * i;
}
alert("i="+i+",num="+num); //输出结果
//-->
</script>
```

上面的例子第 3～4 行定义了两个变量 i 和 num，分别赋值为 1 和 10，随后使用 while 来进行循环，循环的条件是变量 i 小于 5，循环的内容是每次循环让变量 i 加 1，并把当前 num 的值与当前 i 的乘积重新赋值给变量 num，最后用 alert()函数把变量 i 和 num 最终的值以提示框的形式显示。最后运行后结果如图 5.3 所示。

图 5.3　while 循环示例

使用 while 需要注意的一点是，要能在合适的情况下结束循环，否则，程序就会陷入无限的死循环当中去，这是编写程序的过程中非常忌讳的。不仅仅需要设置条件表达式，还需要让条件表达式能够产生作用。看下面的代码段：

```
<script language="JavaScript">
<!--
var i = 1;
var num = 10;
while( i < 5 ){
    num = num * i;          //执行汇总运算
}
//-->
</script>
```

上面的代码段，第 5 行虽然设置了条件表达式“i<5”，但是在循环体内，并没有改变变量 i 的语句。所以变量 i 始终会保持初始值 0，因此条件表达式永远都是 true，所以这个循环将会一直执行。除了使用条件表达式来结束循环以外，还可以使用关键字 break 来跳出循环。看下面的代码：

```
<script language="JavaScript">
<!--
var i = 1;
var num = 10;
```

```
05  while( i < 5 ){
06      num = num * i;       //执行汇总运算
07      if( num > 5 ){
08          break;       //跳出循环
09      }
10  }
11  //-->
12  </script>
```

上面的代码，虽然第 5 行条件表达式始终为 true，不会结束循环，但是可以使用第 8 行的 break 来结束循环。第 7 行使用了 if 语句来对变量 num 的值进行判断，如果大于 100，则通过 break 来结束循环。

5.2.2　do…while 循环

另一个和 while 类似的语句是 do...while 语句。do...whlie 语句也是需要判断条件表达式为 true 或者为 false 来决定是否结束循环，但是和 while 不同的是，do...while 语句不论条件表达式的值如何，都会首先执行一次循环体内的语句或语句块，随后再根据条件表达式来决定是否继续循环。通过查看下面 do...while 语句的语法，更加容易理解：

```
do{
    语句或语句块;
}while( 条件表达式 )
```

可以看出，与 while 语句比起来，除了条件语句的位置发生变化外，还多了一个关键字 do。

【范例 5-5】 下面通过把 5.2.1 小节中的示例改造来理解 do...while 语句。

```
01  <script language="JavaScript">
02  <!--
03  var i = 1;
04  var num = 10;
05  do{
06      i++;                          //i 自增
07      num = num * i;                //执行汇总运算
08  }while( i < 5 )
09  alert("i="+i+",num="+num);        //输出结果
10  //-->
11  </script>
```

上面的代码是按照 do...while 语句的语法把 5.2.1 小节中的示例改造而来的。经过运行后，结果如图 5.4 所示。

图 5.4　do...while 语句示例

可以看到，运行结果与 5.2.1 小节中的 while 语句完全一样。下面通过调整条件表达式，

来对比理解一下 while 和 do...while 语句的区别。

1．while语句代码段

代码如下所示：

```
<script language="JavaScript">
<!--
var i = 1;
var num = 10;
while( i < 1 ){                          //判断条件
    i++;
    num = num * i;                       //汇总
}
alert("i="+i+",num="+num);
//-->
</script>
```

上面的代码段，第 5 行使用了 while 语句，把条件表达式改为了“i<1”，这个表达式的值为 false，因此循环体内的代码不会执行。最后运行结果如图 5.5 所示。

图 5.5　语句对比示例——while 语句

2．do...while语句代码段

代码如下所示：

```
<script language="JavaScript">
<!--
var i = 1;
var num = 10;
do{
    i++;
    num = num * i;                   //汇总
}while( i < 1 )                          //判断条件
alert("i="+i+",num="+num);
//-->
</script>
```

上面的代码段，第 5~8 行使用了 do...while 语句，条件表达式仍然使用“i<1”，表达式的值为 false，但是循环仍然会首先执行一次，运行后结果如图 5.6 所示。

图 5.6　语句对比示例——do...while 语句

说明：使用 do...while 语句需要注意的地方和 while 语句一样，需要设置合适的结束条件或者利用 break，防止产生死循环。

5.2.3　for 循环

在 JavaScript 中，也可以使用 for 语句来实现循环，for 语句的语法结构和 while 很相似。先看一下 for 语句的语法：

```
for( 初始化表达式; 条件表达式; 更新语句 ){
    语句或语句块;
}
```

对比 while 语句的语法，可以看出来，除了关键字由 while 替换为 for 以外，在紧跟关键字 for 的圆括号里，由 while 语句的一个条件表达式，变成了由分号分隔的 3 个独立部分：第一个部分是初始化表达式，用来初始化变量等；第二部分是条件表达式，用于进行循环条件判断；第三部分是更新语句，用来更新某些值，从而改变条件表达式的值，进而控制整个循环。所以，从某种程度上讲，for 语句是 while 语句的一个升级或改进版本。

for 语句的执行顺序如下：

（1）执行初始化表达式。当 JavaScript 遇到 for 语句后，首先执行 for 语句的初始化表达式，初始化表达式通常是声明一个变量并且进行赋值。值得注意的是，初始化表达式只执行一次，不会随着循环的执行而多次执行。

（2）判断条件表达式。

（3）如果条件表达式的值为 false，则结束循环；如果条件表达式的值为 true，则开始循环，执行完循环中的语句后，开始执行（4）。

（4）一次完整的循环最后一步是执行更新语句。更新语句通常用于改变条件语句中变量的值，从而控制循环。

【范例 5-6】 下面的例子，是一个完整的 for 语句的例子。

```
01  <script language="JavaScript">
02  <!--
03  for( var i=0; i < 5; i++ ){
04      document.writeln("i="+i+"<br>");                    //输出结果
05  }
06  //-->
07  </script>
```

上面的代码段，第 3 行先初始化一个变量 i 并赋值为 0，然后当符合条件表达式“i<5”时，显示变量 i 的值，执行完毕后，把变量加 1。执行后结果如图 5.7 所示。

上面的代码段，如果用 while 语句来实现，则如下所示：

```
01  <script language="JavaScript">
02  <!--
03  var i=0;
04  while( i < 5 ){                              //判断条件
05      i++;
```

```
06      document.writeln("i="+i+"<br>");
07  }
08  //-->
09  </script>
```

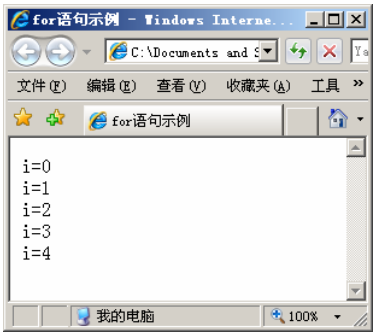

图 5.7　for 语句示例

对比可以看出，使用 while 语句不如使用 for 语句效率高，但是 for 语句常用在含有计数器的程序里，通过计数器变量来作为条件控制循环，因此并不是所有的情况下 for 语句都能替代 while 语句的，在有的情况下，使用 while 语句能够使用其他的条件。比如下面的语句：

```
01  <script language="JavaScript">
02  <!--
03  var f=true;
04  while( f ){
05      f = confirm("继续循环？");                    //判断条件
06  }
07  //-->
08  </script>
```

上面的代码第 5～6 行根据用户在确认提示框单击的按钮，确定是否继续循环，如图 5.8 所示。

图 5.8　循环中产生的确认提示框

上面的这个例子，仍然可以用 for 语句来实现，但是，却要显得比较累赘，看下面的代码：

```
01  <script language="JavaScript">
02  <!--
03  for( var f=true; f==true; f = confirm("继续循环？")){
04  }
05  //-->
06  </script>
```

还有另外一种省略更新语句的写法如下所示：

```
01  <script language="JavaScript">
```

```
<!--
for( var f=true; f==true; ){
    f = confirm("继续循环？");
}
//-->
</script>
```

注意：省略更新语句时，条件语句后的分号不能省略，否则会报错。同样的道理，for 语句的另外两个部分——初始化表达式和条件表达式也可以省略，但是必须要保留分号，否则同样会出现错误。

5.2.4　for…in 循环

在前面章节中介绍过对象的概念，对象可以有很多属性，用来存放信息。for...in 语句就是用来对一个对象的属性进行循环访问的语句。for...in 语句的语法如下：

```
for( 属性名变量 in 对象名 ){
    语句或语句块;
}
```

for...in 语句没有类似 while 或者 for 语句那样的条件表达式来控制循环的结束，for...in 语句循环一直到对象的属性被遍历完毕。因此，使用 for...in 语句的循环次数就是对象的属性个数。使用 for 语句需要自己定义一个属性名变量，然后可以在循环语句里使用这个变量。

【范例 5-7】下面用一个例子来说明 for...in 语句的用法，例子的 HTML 文件为 5-4.html，代码如下所示：

```
<html>
<head>
    <title>for...in 语句访问对象</title>
<script language="JavaScript">
<!--
    //动物（Animal）构造器函数定义
    function Animal(type , sound, food ){
        this.animal_type = type;
        this.animal_sound = sound;
        this.animal_food = food;
    }
    var dog = new Animal("dog", "汪汪", "杂食");        //定义对象
    for( obj_p in dog ){                                //遍历对象
        document.writeln("对象属性"+obj_p+"的值是："+dog[obj_p]+
        "<br>");
    }
//-->
</script>
</head>
<body>
</body>
</html>
```

上面的代码第 7～11 行编写了一个构造器函数 Animal，并为这个对象创建了 3 个属性。第 12 行创建了一个对象并进行了赋值。第 13～15 行使用 for...in 语句，通过自定义的 obj_p 变量来实现对属性的访问，循环显示出新创建的对象的值。运行后如图 5.9 所示。

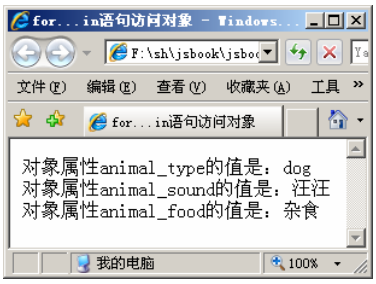

图 5.9　for...in 语句示例

注意：在循环语句里使用的变量 obj_p，用来代替当前循环到的对象属性名称，这个变量的名称可以是自己定义的，并没有其他特殊的要求。

5.2.5　使用 with 实现对属性的访问

处理对象的属性，不仅仅只有 for...in 语句，with 语句也能够实现对属性的访问。with 语句主要可以节省重复输入对象名称。在 with 语句的范围内，可以不用在每个对象属性前面重复地输入对象名称。with 语句的语法如下：

```
with( 对象 ){
    语句或语句块;
}
```

with 关键字后面紧跟着一对圆括弧，圆括弧里面是对象名，然后是一个大括弧，里面是循环的语句或语句块。

在前面的章节里，频繁使用过 document.write 和 document.writeln，如下所示：

```
01  <script language="JavaScript">
02  <!--
03  document.writeln("第一行<br>");
04  document.writeln("第二行<br>");
05  document.writeln("第三行<br>");
06  //-->
07  </script>
```

上面的代码，第 3～5 行向页面显示了一些内容。实际上，document 是 JavaScript 的一个文档对象，write 是这个对象的一个方法，用来显示内容到页面，有时候也会使用 writeln 来实现类似的功能。在使用 document.write 这个语句时，结合 with 语句，可以省略掉 document 这个对象名，看下面的代码：

```
01  <script language="JavaScript">
02  <!--
03  with( document ){
04      writeln("第一行<br>");
```

```
05      writeln("第二行<br>");
06      writeln("第三行<br>");
07  }
08  //-->
09  </script>
```

可以看到，使用了 with 语句，在 writeln 方法前不需要再添加 document 这个对象名字了。

【范例 5-8】 针对 5.2.3 小节里的那个循环访问对象属性的例子，使用 with 语句进行改造后，HTML 文件见 5-5.html，代码如下所示：

```
01  <html>
02  <head>
03      <title>使用 with 访问对象</title>
04  <script language="JavaScript">
05  <!--
06  //动物（Animal）构造器函数定义
07  function Animal(type , sound, food ){
08      this.animal_type = type;
09      this.animal_sound = sound;
10      this.animal_food = food;
11  }
12  var dog = new Animal("dog", "汪汪", "杂食");                    //定义对象
13  with( dog ){                                                  //输出对象属性
14      document.writeln("对象属性 animal_type 的值是:"+animal_type+"<br>");
15      document.writeln("对象属性 animal_type 的值是: "+animal_
        sound+"<br>");
16      document.writeln("对象属性 animal_type 的值是:"+animal_food+"<br>");
17  }
18  //-->
19  </script>
20  </head>
21  <body>
22  </body>
23  </html>
```

上面的代码，第 12～17 行创建了一个名为 dog 的对象，使用了 with 简化对象名的重复编写，在需要输出属性值的时候，直接使用属性名称即可。运行后结果如图 5.10 所示。

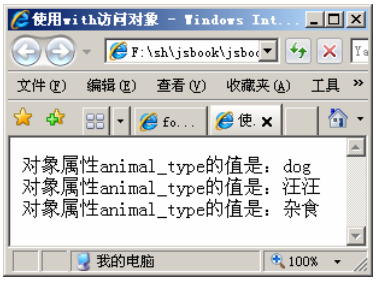

图 5.10　使用 with 访问对象

5.2.6　使用 continue 继续循环

在前面的小节里，接触到 break 语句的内容，break 用来在合适的条件下强制跳出循环。

本小节所要介绍的另一个语句是 continue 语句，具有和 break 类似的终止循环的功能。但 break 是跳出整个循环，不再循环，而 continue 是结束本次循环，跳到下次循环开始的位置。

【范例 5-9】 例如，需要一个程序，用来过滤掉从 1～5 所有整数里 3 的倍数，不让这些数字参与累计运算，那么就需要用到 continue 语句，其代码如下：

```
<script language="JavaScript">
<!--
var num = 0;
for( var i=1; i <=5; i++ ){
    if( i % 3 == 0 ){
        continue;                    //继续循环
    }else{
        num += i;                 //汇总运算
    }
}
alert(num);
//-->
</script>
```

上面的代码，目的是将 1～5 的所有整数累加，但是要过滤掉 3 的倍数。最后运行的结果如图 5.11 所示。

图 5.11　使用 continue 示例

5.3　小　　结

本章主要介绍了 JavaScript 中的两类流程控制语句：判断语句和循环语句。要编写灵活而完善的程序，掌握好这两类语句是非常必要的。在使用这两类语句的同时，笔者根据多年的经验，穿插了很多页面开发的技巧，如避免死循环、及时跳出循环、中断循环等等。希望读者不光能看懂这些程序，还能根据书中的代码，多进行联系，体会流程控制的关键所在。

5.4　习　　题

一、填空题

1．_____语句是跳出整个循环，不再循环，而_____语句是结束本次循环，跳到下次循环开始的位置。

2．_____语句是根据一个固定的表达式的值来进行条件控制的。

二、选择题

1．if…else 可以在语句中再嵌套 if…else 语句吗？（　　）

A　可以，可以无限嵌套

B　不可以，只能单循环

2．以下不是循环语句的是（　　）。

A　while 语句　　B　do…while 语句

C　for 语句　　D　with 语句

三、实践题

1．制作成绩输出表，判断学生的成绩是否大于等于 60，是的话在页面中输出“你及格了”，不是的话，则输出“你不及格”。

【提示】运用本章的条件判断语句和上一章的运算符。

2．输出九九乘法表。

【提示】对初学者来说可能会稍微有一些难度，不过利用循环，可以让代码更简洁。