第3章　控制语句

在JavaScript中，逻辑控制语句用于控制程序的执行顺序，分为条件结构和循环结构。

3.1　条件语句

JavaScript条件判断语句主要包括if语句、if-else语句、if-else if-else语句和switch语句。

1. if语句

if语句是最简单的条件判断语句。

语法格式如下。

```
if(表达式){
    //JavaScript语句;
}
```

表达式返回布尔值，当值是true时，执行JavaScript语句，否则跳过JavaScript语句继续向下执行。

【例3-1】　if条件判断。

参考代码：

```
<!DOCTYPE html>
<html>
    <head><meta charset="utf-8"><title></title></head>
    <body>
        <script type="text/javascript">
            var password = "hello";
            if (password.length < 6) {
                alert("密码长度不足6位,请重新设置!");
            }
        </script>
    </body>
</html>
```

运行结果如图3-1所示。

代码分析：

alert()方法用于显示带有一条指定消息和一个确认按钮的对话框。

password.length < 6判断password字符串的长度是否小于6位，当长度小于6位

图 3-1　if 语句运行结果

时，执行 alert()方法，弹出对话框。

2. if-else 语句

if-else 语句比 if 语句多一种情况。

语法格式如下。

```
if(表达式){
    //JavaScript 语句 1
}
else{
    //JavaScript 语句 2
}
```

表达式返回值为 true 时，执行 JavaScript 语句 1；否则执行 JavaScript 语句 2。

【例 3-2】 if-else 条件判断。

参考代码：

```
<!DOCTYPE html>
<html>
    <head><meta charset="utf-8"><title></title></head>
    <body>
        <script type="text/javascript">
            var score = prompt("请输入你的成绩:");
            if (score >= 60) {
                alert("恭喜你!成功通过了考试!");
            } else {
                alert("很遗憾,继续努力!");
            }
        </script>
    </body>
</html>
```

运行结果如图 3-2 所示。

代码分析：

prompt()方法会弹出一个提示框，用于等待用户输入的数据，返回值类型为字符型。当 score >= 60 进行比较运算时，字符数据先自动转换成数字，然后跟 60 比较。如果返回值是 true，则对话框中显示“恭喜你！成功通过了考试!”；否则显示“很遗憾，继续努力!”。

图 3-2　if-else 条件判断运行过程

3. if-else if-else 语句

if-else if-else 语句用于需要判断多个条件的情况，每个条件对应一段程序。

语法格式如下。

```
if(表达式 1){
    //JavaScript 语句 1
}
else if(表达式 2){
    //JavaScript 语句 2
}
else if
    …
else{
    //JavaScript 语句 n
}
```

if-else if-else 语句首先会执行表达式 1，如果表达式 1 返回 true，执行 JavaScript 语句 1，然后直接跳出这个条件结构，JavaScript 语句 2～JavaScript 语句 n 等都不会被执行。如果表达式 1 返回 false，表达式 2 将会被判断。以此类推，表达式 1～表达式$n-1$都返回 false 时，执行 else 后的 JavaScript 语句 n。

【例 3-3】 使用 if-else if-else 语句实现将百分制成绩转换为五级制成绩，即成绩>=90：优秀；成绩>=80：良好；成绩>=70：中等；成绩>=60：及格；成绩<60：不及格。

参考代码：

```
<!DOCTYPE html>
<html>
    <head><meta charset="utf-8"><title></title></head>
    <body>
        <script type="text/javascript">
            var score = prompt("请输入你的成绩");
                if (score >= 90) {
                    document.write("<h2>你的成绩是:优秀</h2>");
                } else if (score >= 80) {
                    document.write("<h2>你的成绩是:良好</h2>");
                } else if (score >= 70) {
                    document.write("<h2>你的成绩是:中等</h2>");
                } else if (score >= 60) {
```

```
                document.write("<h2>你的成绩是:及格</h2>");
            } else {
                document.write("<h2>你的成绩是:不及格</h2>");
            }
        </script>
    </body>
</html>
```

运行结果如图 3-3 所示。

图 3-3　百分制成绩转换为五级制成绩

4. switch 语句

switch 语句将表达式与一组数据进行比较，当表达式与所列数据相等时，执行相应的代码块。

语法格式如下。

```
switch (表达式) {
    case 常量 1:
        //JavaScript 语句 1;
        break;
    case 常量 2:
        //JavaScript 语句 2;
        break;
        …
    default:
        默认语句;
}
```

当判断条件多于 3 个时，就可以使用 switch 语句，这样可以使程序的结构更加清晰。switch 语句根据一个变量的不同取值执行不同的语句。在执行 switch 语句时，表达式将从上往下与每个 case 语句后的常量做比较。如果相等，则执行该 case 语句后的 JavaScript 语句，如果没有一个 case 语句的常量与表达式的值相等，则执行默认语句。

【例 3-4】　使用 switch 语句进行判断。

参考代码：

```
<!DOCTYPE html>
<html>
    <head><meta charset="utf-8"><title></title></head>
```

```
    <body>
        <script type="text/javascript">
            var weekday = prompt("请输入今天是星期几");
            switch (weekday) {
              case "星期一":
                 document.write("周一,新的开始...");
                 break;
              case "星期二":
              case "星期三":
              case "星期四":
                 document.write("离周末还有好多天,好好努力吧!");
                  break;
              case "星期五":
                 document.write("终于到周末了!");
                 break;
              default:
                 document.write("你输入的是星期几?");
                 break;
            }
        </script>
    </body>
</html>
```

运行结果如图 3-4 所示。

图 3-4　使用 switch 语句判断

代码分析：

switch 语句中用于判断的表达式的值可以是数值、布尔值和字符串。本实例中用于判断的表达式是字符串类型。

根据判断执行相应的 case 语句块，如果代码中有 break 语句则跳出 switch 语句，如果没有则顺序执行下一个 case 语句块。本实例中，如果输入的是“星期二”，则会顺序执行“星期二”“星期三”“星期四”对应的语句块，输出“离周末还有好多天，好好努力吧!”后，执行 break 语句跳出 switch 语句。

3.2　循 环 语 句

JavaScript 中的循环控制语句主要包括 while 循环、do-while 循环、for 循环、for-in 循环以及特殊命令 break、continue。

1. while 循环

语法格式如下。

```
while(条件)
{
    //JavaScript 语句;
}
```

while 循环语句的特点是先判断后执行，当条件为真时，就执行 JavaScript 语句；相反，当条件为假时，则退出循环。

【例 3-5】 使用 while 语句输出递增的数字序列。

参考代码：

```
<!DOCTYPE html>
<html>
    <head><meta charset="utf-8"><title></title></head>
    <body>
        <script type="text/javascript">
            var i = 1;
            while (i <= 10) {
                document.write(i + " ");
                i++;
            }
        </script>
    </body>
</html>
```

运行结果如图 3-5 所示。

图 3-5 while 循环

代码分析：

示例中定义的 i 是循环变量，初始值为 1，每次执行完循环体，i 的值加 1，直到 i 的值大于 10 才结束循环。

2. do-while 循环

语法格式如下。

```
do {
    //JavaScript 语句;
}while(条件);
```

do-while 循环语句表示反复执行 JavaScript 语句，直到条件为假时才退出循环，与 while 循环语句的区别在于，do-while 循环语句先执行后判断。

【例 3-6】　使用 do-while 语句输出递增的数字序列。

参考代码：

```
<!DOCTYPE html>
<html>
    <head><meta charset="utf-8"><title></title></head>
    <body>
        <script type="text/javascript">
            var i = 1;
            do {
                document.write(i + " ");
                i++;
            } while (i <= 10)
        </script>
    </body>
</html>
```

运行结果如图 3-6 所示，与 while 循环的输出结果是一致的。

图 3-6　do-while 循环

代码分析：

例 3-6 的运行结果与例 3-5 是一致的，但 do-while 循环与 while 循环的区别是先执行再判断，这就意味着不管条件是否符合，do-while 循环至少执行一次。

3. for 循环

语法格式如下。

```
for(初始化;条件;增量)
{
    //JavaScript 语句;
}
```

其中，初始化参数设置循环变量的初始值；条件是用于判断循环终止时的条件，若满足条件，则继续执行循环体中的语句，否则跳出循环；增量或减量用于定义循环控制变量在每次循环时如何变化。在 3 个条件之间，必须使用分号(;)隔开。

【例 3-7】　使用 for 循环遍历数组元素。

参考代码：

```
<!DOCTYPE html>
```

```
<html>
    <head><meta charset="utf-8"><title></title></head>
    <body>
        <script type="text/javascript">
            var colors = new Array("yellow", "green", "blue", "red", "orange");
            for (var i = 0; i < colors.length; i++) {
                document.write(colors[i] + "<br/>");
            }
        </script>
    </body>
</html>
```

运行结果如图 3-7 所示。

图 3-7　for 循环遍历数组

代码分析：

示例中循环控制变量 i 初始值为 0，循环条件是小于数组的长度，增量自动累加 1，这样通过 for 循环遍历访问了数组中的每个元素。

4. for-in 循环

for-in 循环常用于数组或对象的遍历。

语法格式如下。

```
for(变量 in 对象){
      //JavaScript 语句;
}
```

其中，“变量”可以是数组元素，也可以是对象。

【例 3-8】 使用 for-in 循环遍历数组元素。

参考代码：

```
<!DOCTYPE html>
<html>
    <head><meta charset="utf-8"><title></title></head>
    <body>
        <script type="text/javascript">
            var colors = new Array("yellow", "green", "blue", "red", "orange");
            for (var i in colors) {
                document.write(colors[i] + "<br/>");
            }
```

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

运行结果与例 3-7 相同，如图 3-8 所示。

图 3-8　for-in 循环遍历数组

【例 3-9】 使用 for-in 遍历 JS 对象。

参考代码：

```
<!DOCTYPE html>
<html>
    <head><meta charset="utf-8"><title></title></head>
    <body>
        <script type="text/javascript">
            var student = {
                name: "张三",
                sex: "男",
                age: 21,
                phone: "13857474113"
            };
            for (key in student) { //获取对象的属性名并保存到变量 key 中
                document.write(key + " : " + student[key] + "<br/>");
            }
        </script>
    </body>
</html>
```

运行结果如图 3-9 所示。

图 3-9　for-in 循环遍历 JS 对象

代码分析：

JSON 是一种轻量级的数据交换格式。JSON 对象由“键(key)/值(value)”对组成，例 3-9 中的 student 是类似于 JSON 的 JS 对象。

在 for-in 循环中，key 遍历 student 对象中所有属性，并通过属性名输出对应的属性值。

5. 中断循环

在 JavaScript 中，有两个特殊的语句用于在循环内部中断循环：break 和 continue。

(1) break：立即退出整个循环。

(2) continue：只退出当前循环，根据判断条件来判断是否进入下一次循环。

【例 3-10】 break、continue 中断循环。

参考代码：

```
<!DOCTYPE html>
<html>
    <head><meta charset="utf-8"><title></title></head>
    <body>
        <script type="text/javascript">
            document.write("break 中断操作:<br/>");
            for (var i = 0; i < 5; i++) {
                if (i == 2) {
                    break;
                }
                document.write("数字是:" + i + "<br/>");
            }
            document.write("<hr/>");
            document.write("continue 中断操作:<br/>");
            for (var i = 0; i < 5; i++) {
                if (i == 2) {
                    continue;
                }
                document.write("数字是:" + i + "<br/>");
            }
        </script>
    </body>
</html>
```

运行结果如图 3-10 所示。

图 3-10　break、continue 中断循环