break和continue语句

break 和 continue 语句用于在循环中精确地控制代码的执行。其中，break 语句会立即退出循环，强制继续执行循环后面的语句。而 continue 语句虽然也是立即退出循环，但退出循环后会从循环的顶部继续执行。请看下面的例子：

```
var num = 0;

for (var i=1; i < 10; i++) {
    if (i % 5 == 0) {
       break;
    }
    num++;
}

alert(num);     //4
```

这个例子中的 for 循环会将变量 i 由 1 递增至 10。在循环体内，有一个 if 语句检查 i 的值是否可以被 5 整除（使用求模操作符）。如果是，则执行 break 语句退出循环。另一方面，变量 num 从 0 开始，用于记录循环执行的次数。在执行 break 语句之后，要执行的下一行代码是 alert()函数，结果显示 4。也就是说，在变量 i 等于 5 时，循环总共执行了 4 次；而 break 语句的执行，导致了循环在 num 再次递增之前就退出了。如果在这里把 break 替换为 continue 的话，则可以看到另一种结果：

```
var num = 0;

for (var i=1; i < 10; i++) {
    if (i % 5 == 0) {
        continue;
    }
```



```
        num++;
    }

    alert(num);     //8
```

break 和 continue 语句都可以与 label 语句联合使用，从而返回代码中特定的位置。这种联合使用的情况多发生在循环嵌套的情况下，如下面的例子所示：

```
var num = 0;

outermost:
for (var i=0; i < 10; i++) {
     for (var j=0; j < 10; j++) {
        if (i == 5 && j == 5) {
            break outermost;
        }
        num++;
    }
}

alert(num);     //55
```

BreakStatementExample02.htm

在这个例子中，outermost 标签表示外部的 for 语句。如果每个循环正常执行 10 次，则 num++ 语句就会正常执行 100 次。换句话说，如果两个循环都自然结束，num 的值应该是 100。但内部循环中的 break 语句带了一个参数：要返回到的标签。添加这个标签的结果将导致 break 语句不仅会退出内部的 for 语句（即使用变量 j 的循环），而且也会退出外部的 for 语句（即使用变量 i 的循环）。为此，当变量 i 和 j 都等于 5 时，num 的值正好是 55。同样，continue 语句也可以像这样与 label 语句联用，如下面的例子所示：

虽然联用 break、continue 和 label 语句能够执行复杂的操作，但如果使用过度，也会给调试带来麻烦。在此，我们建议如果使用 label 语句，一定要使用描述性的标签，同时不要嵌套过多的循环。