第四章　JavaScript初步

本章导读

没有JavaScript的Web世界难以想象。有了JavaScript，静态的Web页面变得动态；有了JavaScript，可以在页面上校验输入数据；有了JavaScript，页面数据就可以根据流程处理。总之，没有JavaScript，Web将失去精彩。

学习目标：

1. 掌握JavaScript数据类型、运算符及其表达式；
2. 掌握JavaScript中引用量和直接量；
3. 掌握JavaScript的基本流程控制语句；
4. 掌握JavaScript中基本的函数定义；
5. 掌握JavaScript中常用对象；

本章目录

JavaScript是一种能让网页更加生动活泼的程序设计语言，也是容易学习且使用方便的语言。如果没有JavaScript，实在很难想象网页会变成什么样。可以说在网页中嵌入脚本语言的想法，大大地促进了Internet。

大概在 1992 年，一家称作 Nombas 的公司开发了一种叫做C--(C-minus-minus，简称 Cmm)的嵌入式脚本语言。Cmm背后的理念很简单：一个足够强大可以替代宏操作(macro)的脚本语言，同时保持与 C和C++足够的相似性，以便开发人员能很快学会。这个脚本语言捆绑在一个叫做 CEnvi 的共享软件中，它首次向开发人员展示了这种语言的威力。Nombas 最终把Cmm 的名字改成了 ScriptEase，原因是后面的部分(mm)听起来过于消极，同时字母 C “令人害怕”。

当 Netscape Navigator 崭露头角时，Nombas 开发了一个可以嵌入网页中的 CEnvi 的版本。这些早期的试验被称为 Espresso Page（浓咖啡般的页面），它们代表了第一个在万维网上使用的客户端语言。而 Nombas 丝毫没有料到它的理念将会成为万维网的一块重要基石。 当网上冲浪越来越流行时，对于开发客户端脚本的需求也逐渐增大。此时，大部分因特网用户还仅仅通过 28.8 kbit/s 的调制解调器连接到网络，即便这时网页已经不断地变得更大和更复杂。而更加加剧用户痛苦的是，仅仅为了简单的表单有效性验证，就要与服务器进行多次地往返交互。设想一下，用户填完一个表单，点击提交按钮，等待了 30 秒的处理后，看到的却是一条告诉你忘记填写一个

必要的字段。 那时正处于技术革新最前沿的 Netscape，开始认真考虑开发一种客户端脚本语言来解决简单的处理问题。当时工作于 Netscape 的 Brendan Eich，开始着手为即将在 1995 年发行的 Netscape Navigator 2.0 开发一个称之为 LiveScript 的脚本语言，当时的目的是在浏览器和服务器（本来要叫它 LiveWire）端使用它。Netscape 与 Sun 及时完成 LiveScript 实现。就在 Netscape Navigator 2.0 即将正式发布前，Netscape 将其更名为 JavaScript，目的是为了利用 Java 这个因特网时髦词汇。Netscape 的赌注最终得到回报，JavaScript 从此变成了因特网的必备组件。

因为 JavaScript 1.0 如此成功，Netscape 在 Netscape Navigator 3.0 中发布了 1.1 版。恰巧那个时候，微软决定进军浏览器，发布了 IE 3.0 并搭载了一个 JavaScript 的克隆版，叫做 JScript（这样命名是为了避免与 Netscape 潜在的许可纠纷）。微软步入 Web 浏览器领域的这重要一步虽然令其声名狼藉，但也成为 JavaScript 语言发展过程中的重要一步。 在微软进入后，有 3 种不同的 JavaScript 版本同时存在：Netscape Navigator 3.0 中的 JavaScript、IE 中的 JScript 以及 CEnvi 中的 ScriptEase。与 C 和其他编程语言不同的是，JavaScript 并没有一个标准来统一其语法或特性，而这 3 中不同的版本恰恰突出了这个问题。随着业界担心的增加，这个语言的标准化显然已经势在必行。

1997 年，JavaScript 1.1 作为一个草案提交给欧洲计算机制造商协会（ECMA）。第 39 技术委员会（TC39）被委派来“标准化一个通用、跨平台、中立于厂商的脚本语言的语法和语义”(http://www.ecma-international.org/memento/TC39.htm)。由来自 Netscape、Sun、微软、Borland 和其他一些对脚本编程感兴趣的公司的程序员组成的 TC39 锤炼出了 ECMA-262，该标准定义了名为 ECMAScript 的全新脚本语言。 在接下来的几年里，国际标准化组织及国际电工委员会（ISO/IEC）也采纳 ECMAScript 作为标准（ISO/IEC-16262）。从此，Web 浏览器就开始努力（虽然有着不同的程度的成功和失败）将 ECMAScript 作为 JavaScript 实现的基础。

第一节　数据类型

在JavaScript中常用的数据类型是数值和字符串。数值用于表达各种类型的数据，比如：正负整数、小数等。和其他高级语言相比，JavaScript在定义变量时不说明其数据类型，而是通过其存在变量中的数据由JavaScript系统自动确定，甚至可以用同一个变量先后保存不同类型的数据，这点和其他很多程序语言有很大差别。另外，在JavaScript中可用typeof()函数判断其数据类型，如例程4-1所示。

例程4-1

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        var myInfo=12345;
第3行        alert(typeof(myInfo));//显示为number
第4行
第5行        myInfo="12345";
第6行        alert(typeof(myInfo));//显示为string
第7行
第8行        var myData1,myData2
第9行        myData1=12345;
第10行       myData2="12345"
第11行       alert(myData1+myData2);//显示为1234512345
第12行       alert(myData2-myData1);//显示为0
第13行       alert(myData2/myData1);//显示为1
第14行       alert("XYZ"-12);//显示为NaN
第15行   </SCRIPT>
```

1、基本数据类型

在例程4-1中，myInfo前后都保存了12345，但由于第5行的12345被加上了双引号，因此被认为是string(字符串)，而第3行被认为是number(数值)。由此可见，在JavaScript中，同一个变量名称(myInfo)既可以存数值，也可以存字符串。另外，虽然myInfo存字符串但是，不能对myInfo加上双引号，因为myInfo是变量或引用量，使用的是其所代表的值。如果给myInfo加上双引号，则表示由这6个字符构成的字符串本身，而不是其所代表的内容。

在例程4-1中第8行到第14行显示数值型与字符串型混合运算时的情况。第11行相当于两个字符串拼接，数值型数据自动被转换为字符串型，而第12行则表示两个数值型数据相减，即自动将字符串型数据转换为数值型，第13行与之同理。由于字符串有加法运算，相当于字符串拼接，当字符串型数据与数值型数据进行加法运算时，将自动将数值型数据转换成字符串型，而进行其他运算，则将字符串型数据转换为数值型。如果是减法、乘法或者除法，且字符串开始字符不是数字，则不能转换为数值型，系统将显示为NaN(Not a Number的简写)，如第14行所示。

默认规则不如显规则让人更明白清晰。JavaScript可通过parseInt()、parseFloat()、Number()函数将字符串转换为数值型数据，也可以通过DataType.toString()将数值转换成字符串。如例程4-2所示。

例程4-2

```
<SCRIPT LANGUAGE="JavaScript">
        var myInfo1=1234.12,myInfo2="1234.12";

        alert(myInfo1+myInfo2);//字符串拼接，1234.121234.12
        alert(myInfo2+myInfo1.toString());////字符串拼接，1234.121234.12

        alert(Number(myInfo2)+myInfo1);//2468.24
        alert(parseInt(myInfo2)+myInfo1);//2468.12,parseInt()转换为整数
        alert(parseFloat(myInfo2)+myInfo1);//2468.24
</SCRIPT>
```

在例程4-2中，第4行采用默认规则将数值型转换成字符串拼接，第5行是将数值型转换成字符串后拼接，是明白无误的转换形式完成。第7行到第9行都是字符串型数据转换成数值型。注意parseInt()将被转换成整数，parseFloat()则可以是实数。

将Number型数据转换成String时(DataType.toString())，虽然一般不必传递参数，但仍然可以传递进制参数，将将之转换为指定的进制，如例程4-3所示。

例程4-3

```
<SCRIPT LANGUAGE="JavaScript">
        var num = 12;
        alert(num.toString()); //"12"
        alert(num.toString(2)); //"1100"
        alert(num.toString(8)); //"14"
        alert(num.toString(10)); //"10"
        alert(num.toString(16)); //"c"
</SCRIPT>
```

例程4-3中，将数值型num变量转换成了需要的进制。

字符串型数据与数值型数据混合运算在Web页面开发中比较常见。例程4-4是实现图4-1所示效果，即单击图片时可以顺序载入下一张图片，实现有限空间更多内容呈现。例程中图片来自北京大学网站(http://www.pku.edu.cn)，其图片编号规则是1-10，扩展名为gif，亦可根据需要变换成其他位置。

图4-1　单击图片顺序切换图片

例程4-4

```
<HTML>
    <HEAD>
```

```
第3行                    <TITLE>单击图片切换</TITLE>
第4行                    <Script type="Text/JavaScript" language="JavaScript" src="./jquery-1.4.2.min.js"></Script
>
第5行                    <META http-equiv="Content-type" content="text/html; charset=gb2312">
第6行            </HEAD>
第7行
第8行            <BODY>
第9行                    <IMG src="http://www.pku.edu.cn/images/index/main_img/1.gif" style="width:100px" ImgNo
="1">
第10行                   <SCRIPT LANGUAGE="JavaScript">
第11行                           $("IMG").click(function(){
第12行                                   var webSite="http://www.pku.edu.cn/images/index/main_img/";
第13行
第14行                                   var currentImgNo=$(this).attr("ImgNo");//取出当前图片编号
第15行                                   var nextImgNo=parseInt(currentImgNo)+1;//计算出下一个图片编号
第16行
第17行                                   var imgSRC=webSite+nextImgNo.toString()+".gif";//合成IMG标志的src属性
第18行
第19行                                   $(this).attr("src",imgSRC);//改变click图片的src属性
第20行                                   $(this).attr("ImgNo",nextImgNo);//修改ImgNo属性，保存当前编号
第21行                           });
第22行                   </SCRIPT>
第23行           </BODY>
第24行   </HTML>
```

从例程4-4第9行可知，变化的是图片的编号，其前后两部分都不会发生改变，即IMG标志的src属性有三部分构成，图片位置、图片编号以及图片扩展名，其中图片编号会发生改变。在程序中，用webSite变量代表图片位置，用currentImgNo代表正在展示的图片编号，将会展示的下一个图片编号用nextImgNo代表，其规则是currentImgNo加上1。由于currentImgNo来自IMG标志的ImgNo属性，通过attr()函数取出后，其内容为字符串型数据，因此通过parseInt()函数将之转换为整数。第17行是计算出IMG 标志的src属性内容，并将结果保存在imgSRC中。第19行修改IMG标志的src属性，第20行将当前展示图片的编号修改为nextImgNo。

这个程序在实际运行中存在问题，当图片编号大于10如11，12时，图片11.gif、12.gif已经不存在，显示就将出现问题，网页出现找不到图片现象。如何解决这个问题呢？通过条件语句控制。当图片编号大于10时，自动将图片编号改为1，从头开始，或者弹出对话框，提醒“已经到最后一张图片！”请参看本章之“流程控制语句”。

除了基本的字符串和数值型外，JavasScript还有布尔型等其他数据类型，将在本章其他位置讲解。

2、复合数据类型

JavaScript支持复合数据类型-对象(Object)，前面讲过的Math就是一种对象，在对象中可以集成数据和操作数据的函数(又称方法)，对象中的数据和函数依赖对象而存在。在JavaScript中，内置有多种对象，比如：数组对象(Array)、字符串对象(String)、数学对象(Math)、日期对象(Date)等；同时，JavaScript还支持自定义对象。

⑴ 数组对象

数组定义如例程4-5第2行所示，从中可以看出，一个变量名中有多个数据。通过变量名称后方括号及括号内的数字可以读写(R&W)数组中的一个成员，第4行取得数组成员的值(Read)，第6行是改变数组成员的值(Write)。

例程4-5

```
第1行    <Script Language="JavaScript">
第2行            var arrData=[12,23,32,45,54,999];//方括号定义数据的简捷方式
第3行            alert(arrData[2]);//显示32，注意数组成员的编号从0开始
第4行
```

```
第5行          arrData[2]=3232;
第6行          alert(arrData[2]);//3232
第7行
第8行          arrData[2]=arrData[5];
第9行          alert(arrData[2]);//显示999
第10行
第11行         arrData[7]=1001;//数组成员数量扩张到8，编号0-7
第12行
第13行         alert(arrData[7]);//1001
第14行         alert(arrData[6]);//undefined,未定义，数组成员6没有定义
第15行
第16行         var arrLength=arrData.length;//length是数组的属性，为数组长度
第17行         alert(arrLength);//注意：数组长度已从开始的6变为8
第18行
第19行         arrData.length=4;//减小length大小，将删除其后尾部数据，在本例中删除编号4-7
第20行         alert(arrData);//显示为12,23,999,45
第21行
第22行         arrData.length=9;//减小length大小，将删除其后尾部数据，在本例中删除编号4-7
第23行         alert(arrData[8]);//从编号4开始到8为止的成员值都为undefined。
第24行     </Script>
```

数组成员的编号从0开始，如“var arrData=[12,23,32,45,54,999]”，其数组长度为6，其编号为从0开始，到5结束，通过arrData[1]访问编号为1的成员，相当于数组中的第2个成员。如果对超过最大编号的数组成员赋值，将自动扩增数组成员数量，如第14行所示，在此之前，数组成员仅有6个，但“arrData[7]=1001;”相当于访问第8号(编号为7)成员，数组成员的数量扩增到8，即数组长度为8。

length是数组对象的属性，其值为数组成员的数量即长度。如果对该属性赋值，可以扩增或缩短数组长度。缩短时自动截取超长度成员(如第20行所示)，扩增时新成员的值为undefined(如第22、23行所示)。

⑵ 自定义对象

除了内置对象外，还可以自定义对象。如例程4-6所示。

例程4-6

```
第1行      <SCRIPT LANGUAGE="JavaScript">
第2行          //定义myObj为一个自定义对象
第3行          var myObj={studentName:"张珊",studentID:"111011101"};
第4行          alert(myObj.studentName);//显示：张珊
第5行          myObj.studentName="李斯";//改变myObj对象的studentName属性值
第6行          alert(myObj.studentName);//显示：李斯
第7行      </SCRIPT>
```

第三行{studentName:"张珊",studentID:"111011101"}就是自定义对象的一种方法，其中studentName是属性名称，"张珊"是属性值，构成一个“键值对”(Key/Value)，其间用冒号分割。多个属性"键值对"(Key/Value)之间用逗号分割。注意：自定义对象用花括号，而数组采用方括号。

例程4-7是自定义对象的一个应用，类似应用在jQuery中还有很多。

例程4-7

```
第1行      <SCRIPT LANGUAGE="JavaScript">
第2行          $("Div").css({width:"400px",height:"80px","background-color":"red"});
第3行      </SCRIPT>
```

自定义对象用途很广泛，例程4-8是采用自定义对象描述平面坐标并计算其距离。

例程4-8

```
第1行  <Script Language="JavaScript">
第2行          var pointFirst={X:1,Y:2};
第3行          var pointSecond={X:7,Y:10};
第4行          //两点横坐标差值的平方与两点纵坐标差值的平方之和的平方根
第5行          var tempVal=Math.pow((pointFirst.X-pointSecond.X),2)+Math.pow((pointFirst.Y-pointSecond.Y),2);
第6行          var distance=Math.sqrt(tempVal);
第7行          alert(distance);//显示为10
第8行  </Script>
```

第二节　流程控制语句

流程控制语句分为两大类：条件分支和条件循环。条件分支语句的功能是满足某种条件执行某些语句，否则执行另外的语句。条件循环语句的功能是满足某种条件下，则重复执行某种语句。

1、if分支语句

如例程4-9所示。第6行if语句的功能是判断b*b-4*a*c是否大于等于0。如果if后括号内的结果为真(true)，则执行紧随其后的语句，如果不满足(false)，则执行else后的语句。if语句可以没有else部分，此时则满足条件执行紧随其后的语句，否则则执行if语句体之后的语句，其执行流程如图4-2所示。

例程4-9

```
第1行  <SCRIPT LANGUAGE="JavaScript">
第2行          /*一元二次方程的判别式，a,b,c分别为系数*/
第3行          var a=1;
第4行          var b=2;
第5行          var c=-3;
第6行          if ((b*b-4*a*c)>=0){
第7行                  alert("有实数根!");//if后条件语句满足时执行
第8行          }else{
第9行                  alert("无实数根!");//if后条件语句不满足时执行
第10行         }
第11行 </SCRIPT>
```

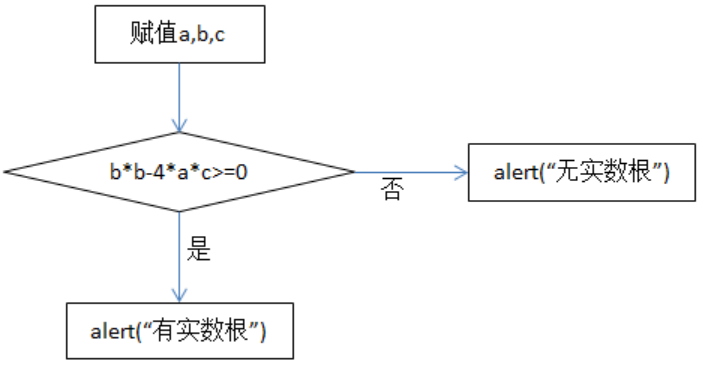

图4-2　if语句执行示意图

if语句可以没有else部分，此时则满足条件执行紧随其后的被if所控制的语句，否则则执行if语句体之后的语句，如例程4-10所示，其流程如图4-3所示。将例程4-4代码做例程4-10修改后，就可以从头到尾循环展示。如果在加上时间控制，即定时更换照片，就更加具有实用性。该功能请阅读本章其他小节。

例程4-10

```
第1行  <SCRIPT LANGUAGE="JavaScript">
```

```
第2行          $("IMG").click(function(){
第3行                  var webSite="http://www.pku.edu.cn/images/index/main_img/";
第4行
第5行                  var currentImgNo=$(this).attr("ImgNo");//取出当前图片编号
第6行                  var nextImgNo=parseInt(currentImgNo)+1;//计算出下一个图片编号
第7行                  if(nextImgNo>10){
第8行                          nextImgNo=1;//如果大于10，则置为1，从头开始展示
第9行                  }
第10行
第11行                 var imgSRC=webSite+nextImgNo.toString()+".gif";//合成IMG标志的src属性
第12行
第13行                 $(this).attr("src",imgSRC);//改变click图片的src属性
第14行                 $(this).attr("ImgNo",nextImgNo);//修改ImgNo属性，保存当前编号
第15行         });
第16行 </SCRIPT>
```

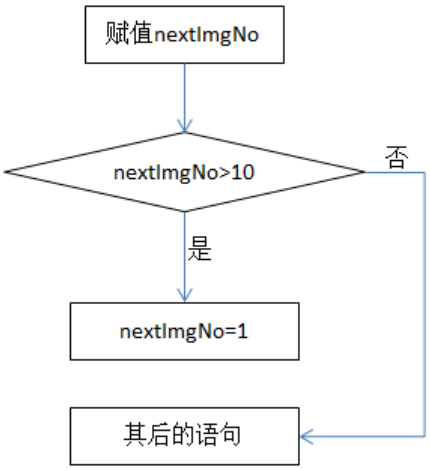

图4-3　if语句执行示意图

例程第7-9行用于控制编号是否超过限制，当编号超过限制时，图片编号设置为1，即从头开始，其含义是当imgNo大于10时执行第8行代码将imgNo设置为1；当编号没有大于10时，则if语句体内的代码不被执行。

2、switch分支语句

除了if分支语句外，还有switch分支语句。当有多个平行条件时，适合使用switch语句，如例程4-11所示。例程4-11是将阿拉伯数字转换为汉字数字，其形式为如果数字为1，转换为壹；如果数字为2，转换为贰，……，依次类推。每一个条件之间没有制约关系，是一种平行关系且可穷尽罗列，这种形式的分支语句，适合用switch。当然，switch语句可以用if表达，但不够简练。

例程4-11

```
第1行  <SCRIPT LANGUAGE="JavaScript">
第2行          var myNum=8;//myNum为1位正整数
第3行          switch(myNum){
第4行                  case 0:
第5行                          alert("零");
第6行                          break;
第7行                  case 1:
第8行                          alert("壹");
第9行                          break;
第10行                 case 2:
第11行                         alert("贰");
```

```
第12行                    break;
第13行                case 3:
第14行                    alert("叁");
第15行                    break;
第16行                case 4:
第17行                    alert("肆");
第18行                    break;
第19行                case 5:
第20行                    alert("伍");
第21行                    break;
第22行                case 6:
第23行                    alert("陆");
第24行                    break;
第25行                case 7:
第26行                    alert("柒");
第27行                    break;
第28行                case 8:
第29行                    alert("捌");
第30行                    break;
第31行                case 9:
第32行                    alert("玖");
第33行                    break;
第34行                default:
第35行                    alert(myNum+"不符合要求!");
第36行            }
第37行    </SCRIPT>
```

switch分支语句的语法形式如下所示，其中n只能为数值(且常为整数)或字符串，default为默认执行语句，即罗列的条件都不满足时，执行该语句，default部分可以省略。

一般说来，每个case后都有break语句，表示该case执行到此处为止。如果省略break，将执行下一个case后的语句。如省略例程4-11第30行的break，则将在执行第29行，继续执行第32行。这是初学者较为常犯的错误。switch执行过程可以参考图4-4所示，如果条件i省略了break，则将执行“语句i+1”。

```
switch(n){
    case 1:
        //执行代码块 1
        break;
    case 2:
        //执行代码块 2
        break;
    default:
        //n 与 case 1 和 case 2 不同时执行的代码
}
```

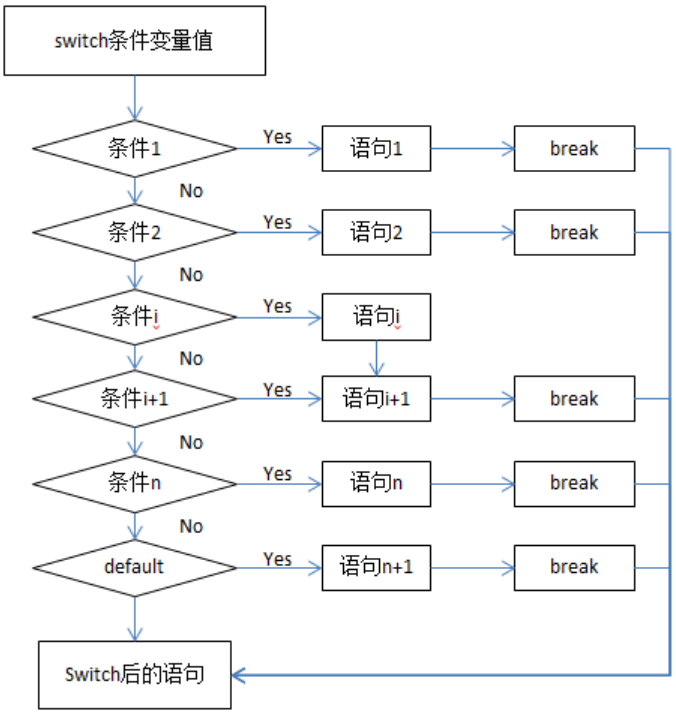

图4-4　switch语句执行示意图

每个case后的值，通常为常量(直接量)，如果是字符串常量，也必须用引号包围。每个case结束后，必须用英文冒号(:)结束，不能是分号(;)。

3、for循环

循环是程序设计语言的重要组成部分，循环有多种形式，如for循环，while循环，do while循环。求1到100的和是一个很常见的例子，就可以用for循环实现，代码如例程4-12所示。

例程4-12

```
<Script Language="JavaScript">
    //求1-100的和
    var sum=0;//sum用于保存求和的结果
    for(var i=1;i<=100;i++){//从1-100的表达
        sum=sum+i;
    }
    alert(sum);//显示5050
</Script>
```

for循环的格式如下，其流程图如图4-5所示。语句1是循环开始执行的语句，语句2是循环的条件，当满足语句2所列条件时执行循环，否则退出循环，语句3是每次循环完毕后所执行的语句，通常用于修改循环的变量，例程4-12即将i的值增大1。

```
for (语句1;语句2;语句3){
    //循环体内执行代码
}
```

在例程4-12中，for(var i=0;i<=100;i++)中的i称为循环变量；i<=100是循环的条件，当i的值小于或等于100时，执行循环体内的语句，否则退出循环；i++是循环变量自增1，即每次增大1。当for循环被执行时，首先执行语句1，然后执行语句2，判断是否满足条件，如果满足条件执行循环体内的执行代码，如果不满足，则跳出循环。循环体内的语句被执行后，然后执行语句3，然后再执行语句2判断是否满足条件，如此依次循环。直到不满足语句2的条件为止。

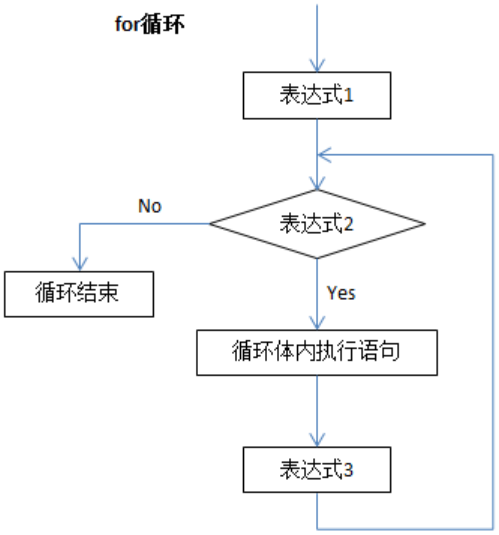

图4-5　for循环示意图

4、while循环语句

条件循环语句又称循环语句，当满足某种条件时，即循环。例程4-13是一个循环，其while()括号内为true时，满足循环条件，循环。在循环体内，仅有break;语句。当循环语句遇到break语句时，将终止循环，并跳到循环体外执行。因此，例程4-13仅被执行了一次，似乎不算循环，但是当删除break;语句后，将无限循环，成为死循环。

例程4-13

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        while(true){
第3行            break;
第4行        }
第5行    </SCRIPT>
```

例程4-14较例程4-13略微复杂。第2行让变量cycleNum等于1，然后进入循环。alert()函数显示cycleNum的值，然后cycleNum增大1。如果cycleNum不大于10，则继续循环。然后一次就是alert()显示cycleNum的值，cycleNum增大1，判断是否大于10。当cycleNum等于10时，alert()显示为10，cycleNum变为11，此时if语句满足cycleNum>10，因此执行break;终止循环。

例程4-14

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        var cycleNum=1;
第3行        while(true){
第4行            alert(cycleNum);//显示1,2,3,4,……,10
第5行            cycleNum++;//每次循环cycleNum增大1;
第6行
第7行            //当cycleNum大于10，则break，终止循环
第8行            if (cycleNum>10){
第9行                break;
第10行           }
第11行       }
第12行   </SCRIPT>
```

图4-6是例程4-14的流程图，能更清楚了解代码执行过程。

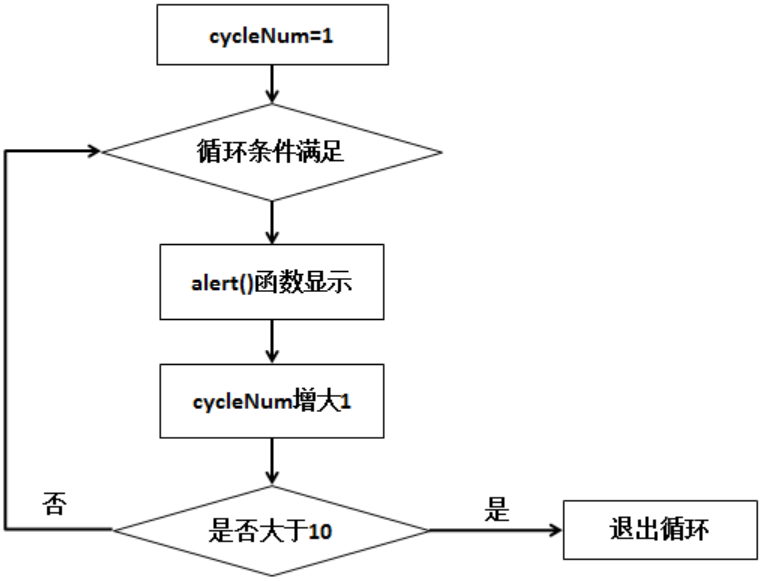

图4-6　例程4-8流程图

例程4-14中的if条件语句即第8行到第10行完全可以删除，但需要做例程4-15这样的简单变化。

例程4-15

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        var cycleNum=1;
第3行
第4行        //cycleNum小于等于10时循环，否则不进入循环体
第5行        while(cycleNum<=10){
第6行
第7行            alert(cycleNum);//显示1,2,3,4,……,10
第8行            cycleNum++;//每次循环cycleNum增大1;
第9行
第10行       }
第11行   </SCRIPT>
```

第5行的功能是首先判断cycleNum是否小于等于10，如果满足条件则循环，否则不循环或者终止循环。很明显，例程4-15的代码相对例程4-14更加简洁，这也是编程时更加一般的形式，流程图如图4-7所示。

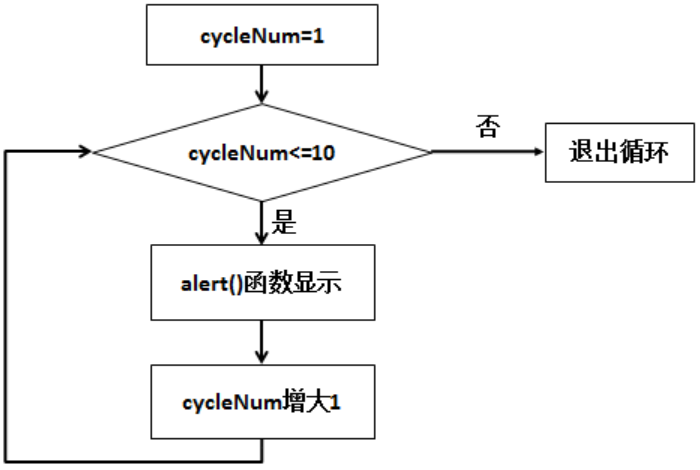

图4-7　例程4-15流程图

例程4-15只是简单显示数字，例程4-16能完成求1到100的和。

例程4-16

```
<HTML>
    <HEAD>
        <TITLE>求1-100的和</TITLE>
        <Script type="Text/JavaScript" language="JavaScript" src="./jquery-1.4.2.min.js"></Script>
        <META http-equiv="Content-type" content="text/html; charset=gb2312">
    </HEAD>

    <BODY>
        <SCRIPT LANGUAGE="JavaScript">
            var cycleNum=1;
            var sum=0;//保存每次计算出的和

            //cycleNum小于等于100时循环，否则不进入循环体
            while(cycleNum<=100){
                //sum与cycleNum相加，而cycleNum从1变化到100
                sum=sum+cycleNum;
                cycleNum++;//每次循环cycleNum增大1；
            }
            alert(sum);//显示为5050
        </SCRIPT>
    </BODY>
</HTML>
```

有了循环，操纵网页就更加具有灵活性，如例程4-17所示，其运行效果如图4-8所示。

例程4-17

```
<HTML>
    <HEAD>
        <TITLE>自动添加编号</TITLE>
        <Script type="Text/JavaScript" language="JavaScript" src="./jquery-1.4.2.min.js"></Script>
        <META http-equiv="Content-type" content="text/html; charset=gb2312">
    </HEAD>

    <BODY>
        <H2>春秋战国</H2>
        <H2>秦汉</H2>
        <H2>唐宋</H2>
        <H2>元明清</H2>
        <SCRIPT LANGUAGE="JavaScript">
            var objChapter=$("H2");//变量保存的是所有H2标记
            var chapterCount=objChapter.size();//取得H2标记的数量

            var cycleNum=0;
            while(cycleNum<chapterCount){
```

```
第19行                    var originalContent=objChapter.eq(cycleNum).html();//取出指定cycleNum编号的内容
第20行                    var chapterNo="第"+(cycleNum+1).toString()+"章 ";//形成类似第1章这样的编号
第21行                    objChapter.eq(cycleNum).html(chapterNo+originalContent);//将新的内容写入指定编号
第22行                    cycleNum++;
第23行                }
第24行            </SCRIPT>
第25行        </BODY>
第26行    </HTML>
```

第1章 春秋战国

第2章 秦汉

第3章 唐宋

第4章 元明清

图4-8 例程4-17效果图

5、do while循环语句

do while循环是while循环的变体，该循环至少会执行一次循环体内语句，然后检查条件是否为真，如果条件为真，则继续循环直到条件不为真。和while循环相比，while是先检查是否符合循环条件，如果符合则执行循环体内语句，如果不符合，则直接跳过或结束循环；而do while则至少会执行一次循环体内，然后根据条件决定是否继续循环。while循环与do while循环对比可参见如图4-9所示，例程4-18是do while循环的一个应用。

```
do{
    //需要执行的代码
}while (条件);
```

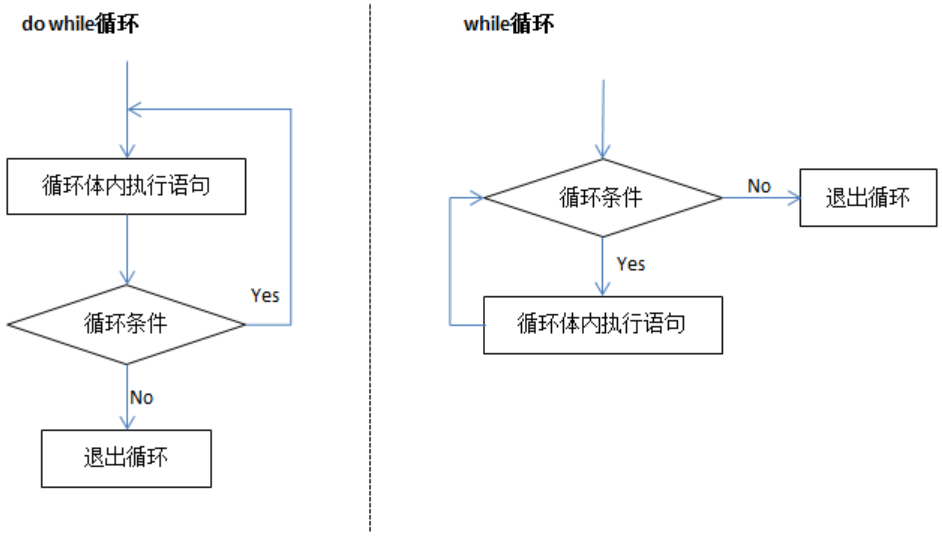

图4-9 do while与while循环示意对比图

例程4-18

```
第1行    <Html>
第2行        <Head>
第3行            <Title>do while循环</Title>
```

```
第4行        </Head>
第5行
第6行        <Body>
第7行            <Script Language="JavaScript">
第8行                //求1-100的和
第9行                var cycleNum=1,sum=0;
第10行               do{
第11行                   sum=sum+cycleNum;
第12行                   cycleNum++;
第13行               }while (cycleNum<=100);
第14行               alert(sum);//显示5050
第15行           </Script>
第16行       </Body>
第17行   </Html>
```

第三节　再说变量

例程4-19

```
第1行    <Script Language="JavaScript">
第2行        for(var i=0;i<=100;i++){
第3行            var sum=0;
第4行            sum=sum+i;
第5行        }
第6行        alert(sum);//显示为100
第7行    </Script>
```

变量是计算机语言的重要组成部分。在JavaScript中，变量的申明格式如下:

```
var varName;
```

其中，varName是变量的名称，可根据自己需要自己命名，如studName、studBirthday等， 但不能与JavaScript中已有的名称重复，如不能命名为alert，var等。 关键字var说明其后的varName为新的变量。如：

```
var myOld;
```

其中，myOld是变量的名称。

变量用于保存数据，等号(=)可给变量存入数据，称之为变量赋值，如：

```
myOld=20;
```

就是给变量myOld存入数据20。例程4-20包含了变量申明、变量赋值以及变量引用等。

例程4-20

```
第1行    <HTML>
第2行        <HEAD>
第3行            <TITLE>JavaScript变量</TITLE>
第4行        </HEAD>
第5行        <BODY>
```

```
第6行                    <SCRIPT LANGUAGE="JavaScript">
第7行                        var myOld;
第8行                        myOld=20;
第9行                        alert(myOld);
第10行                   </SCRIPT>
第11行               </BODY>
第12行           </HTML>
```

执行例程4-12将弹出窗口，并显示20，其为变量myOld的值。

在例程4-12中，第7行申明一个变量，第8行给变量存入一个数，第9行将该数显示出来。 在JavaScript中，通常要求变量“先申明后使用”，这是一个良好的工程习惯。 在很多其他语言中，甚至是必须如此，如C、C++等。

在申明变量的同时，还可以给变量赋值，其格式如下：

```
var varName=value;
```

如:

```
var myOld=20;
```

变量中保存的值，可以被改变。如例程4-13所示。

例程4-21

```
第1行    <HTML>
第2行        <HEAD>
第3行            <TITLE>变量与字符型直接量</TITLE>
第4行        </HEAD>
第5行
第6行        <BODY>
第7行            <SCRIPT LANGUAGE="JavaScript">
第8行                var myOld=18;
第9行                alert(myOld);
第10行
第11行               myOld=19;
第12行               alert(myOld);
第13行           </SCRIPT>
第14行       </BODY>
第15行   </HTML>
```

在例程4-13中，myOld初为18，其后在第11行被改变为19。正因为变量存入的值可以变化， 所以被称为变量。

第四节　引用量和直接量

比较例程4-12第9、10两行，发现alert函数后的ABC一个有引号，而另外一个没有引号；显示的内容也不相同，第9行显示30，而第10行直接显示ABC三个英文字母。在程序设计中，称第9行的ABC为引用量，第10行的"ABC"为直接量。引用量是用符号(比如变量名称)代表其值，而直接量则直接应用其值。在JavaScript中，引用量一定不能有引号，而直接量则可能有引号。当直接量是一个或多个字符时， 必须有引号，而直接量是数值时，则无需引号，直接写上数值即可，如例程4-13所示。

例程4-22

```
第1行    <HTML>
第2行        <HEAD>
```

```
第3行                <TITLE>直接量和应用量</TITLE>
第4行            </HEAD>
第5行
第6行            <BODY>
第7行                <SCRIPT LANGUAGE="JavaScript">
第8行                    var ABC=30;
第9行                    alert(ABC);
第10行                   alert("ABC");
第11行               </SCRIPT>
第12行           </BODY>
第13行       </HTML>
```

在例程4-13中，第8行显示数值型直接量5，5的两侧不能有引号，第9行是两个直接量相加；第10行因为在5和2两侧加上引号，因此作为字符型直接量，此时加号的功能是将两个字符串拼接在一起，所以显示为52。

例程4-23

```
第1行    <HTML>
第2行        <HEAD>
第3行            <TITLE>数值和字符直接量</TITLE>
第4行        </HEAD>
第5行
第6行        <BODY>
第7行            <SCRIPT LANGUAGE="JavaScript">
第8行                alert(5);
第9行                alert(5+2);
第10行               alert("5"+"2");
第11行           </SCRIPT>
第12行       </BODY>
第13行   </HTML>
```

变量不仅可以存直接量，也可以存入引用量，如例程4-14所示。第8行申明变量myOld的值为直接量18，第9行申明变量yourOld的值为引用量myOld的值。在例程4-14中，第10、11行都显示为18。

例程4-24

```
第1行    <HTML>
第2行        <HEAD>
第3行            <TITLE>变量与字符型直接量</TITLE>
第4行        </HEAD>
第5行
第6行        <BODY>
第7行            <SCRIPT LANGUAGE="JavaScript">
第8行                var myOld=18;
第9行                var yourOld=myOld;
第10行               alert(myOld);
第11行               alert(yourOld);
第12行           </SCRIPT>
第13行       </BODY>
第14行   </HTML>
```

第五节　运算符

计算机程序设计语言中的运算符与我们从小就学习的数学中的运算符有很多相似，如算术中的加减乘除在程序设计语言中几乎具有相同的功能，大于小于等用于关系运算的符号也与算术中的符号相同或者相似，功能也相似。JavaScript的运算符与C、C++、Java相似。

在JavaScript中，运算符可以分为赋值运算符、算术运算符、关系运算符、逻辑运算符等，其中赋值运算符用于给变量赋值，算术运算符用于数值计算，关系运算符用于比较大小相等关系，逻辑运算符又称布尔运算，其运算结果为真(true)或假(false)。

1、赋值运算符

赋值运算符用于给变量赋值，最常见的赋值运算符是等号(=)，例程4-25是一些常见的赋值运算形式。计算机语言中的等号与数学中的等号含义不仅相同。在数学中，X=X+11是不成立的，等号两边消除X后就是0=11。在计算机语言中，X=X+11应理解为将等号右边的计算结果赋值给等号左边的变量，在X=X+11中，就是取出变量X的值并加上11，将其计算结果赋值给X，相等于给X加上11。仅仅写成X+11并没有改变X的值，仅仅是X的值加上11，但没有进行赋值运算，X的值保持不变。

例程4-25

```
第1行   <HTML>
第2行       <HEAD>
第3行           <TITLE>赋值运算</TITLE>
第4行       </HEAD>
第5行       <BODY>
第6行           <SCRIPT LANGUAGE="JavaScript">
第7行               var X=18;//申明变量并赋初值
第8行               X=X+11;//X值变为29
第9行               X=X*2;//X值变为58
第10行              X*=2;//X值变为116
第11行              alert(X);//X值维持不变
第12行              alert(X*2)//X值维持不变，但alert函数显示为232
第13行          </SCRIPT>
第14行      </BODY>
第15行  </HTML>
```

在例程4-25的第9、10两行都是将X值乘上2，第10行是第9行的简写形式且运算速度更高。表1是赋值运算符及其简写形式。

表1：JavaScript的赋值运算符

运算符	描述	案例	等价于	结果
=	赋值运算	X=Y		X=5
+=	加法赋值运算	X+=Y	X=X+Y	X=20
-=	减法赋值运算	X-=Y	X=X-Y	X=10
*=	乘法赋值运算	X*=Y	X=X*Y	X=75
/=	除法赋值运算	X/=Y	X=X/Y	X=3
%=	余数赋值运算	X%=Y	X=X%Y	X=0
<<=	左移赋值运算	X<<=Y	X=X<<Y	X=352
>>=	有符号右移赋值运算	X>>=Y	X=X>>Y	X=0
>>=	有符号右移赋值运算	X>>>=Y	X=X>>>Y	X=0
本表中X初值为15，Y的初值为5，各行互不影响。				

例程4-26是赋值时常见的错误。当执行赋值语句时，JavaScript首先执行等号右边的式子。在例程4-26中第2行，首先取出X的值然后加上10，但是此时X的值未知，不能与10进行运算。因此，其后的alert(X)被显示为NaN(Not a Number的简写)。第5行也是错误，当执行Y++时，首先要知道X的初始值，不然不能执行加1运算。当一个新的变量出现时，系统并不会默认给变量赋值，在此处更不会默认赋值为0。在第5行前增加"var Y=0;"，则该语句不会出现错误。

例程4-26

```
<Script Language="JavaScript">
    var X=X+10;//这是错误!
    alert(X);//显示NaN，NaN是Not a Number的简写

    Y++;//报错，提示信息是Y未定义
    alert(Y);
</Script>
```

2、算术运算符

算术运算符用于数值计算，如X=5+2，其中的加号就是算术运算符，常见的算术运算符如表2所示。在算术运算符中%用于求余数，常用于判断能否整除，求余数等，例程4-27“数制转换”就是%的一个典型应用。

表2：JavaScript的算术运算符

运算符	描述	案例	结果
+	加法	x=y+5	x=16
-	减法	x=y-5	x=6
*	乘法	x=y*5	x=55
/	除法	x=y/5	x=2.2
%	求余数	x=y%5	x=1
++	累加	x=++y x=y++	x=12 x=11
--	递减	x=--y x=y--	x=10 x=11
本表中y=11。重点关注++，--在变量前后的区别。			

例程4-27

```
<HTML>
    <HEAD>
        <TITLE>数制转换</TITLE>
    </HEAD>
    <BODY>
        <SCRIPT LANGUAGE="JavaScript">
            //十进制转换为八进制
            var decNum=111;//decNum为十进制正整数
            var Result="";//变量Result保存转换结果
            while(decNum>0){
                var remainder=decNum%8;//remainder变量保存余数
                Result=remainder.toString()+Result;
                decNum=(decNum-remainder)/8;
            }
            alert(Result);//显示157
        </SCRIPT>
    </BODY>
</HTML>
```

累加和递减运算符在变量前或后其含义有所区别，如例程4-28所示。当++置于变量之后时，先去的变量的值作为该部分表达式的值，

然后执行累加，如第3行所示，即变量值“后变”，当置于变量之前时，先执行累加，如第5行所示。递减运算符与累加运算符类似。

例程4-28

```
第1行  <SCRIPT LANGUAGE="JavaScript">
第2行      var X=100;
第3行      alert(X++);//显示100
第4行      alert(X);//显示101
第5行      alert(++X);//显示102
第6行      alert(X);//显示102
第7行  </SCRIPT>
```

累加和递减运算符还常用于循环，如例程4-29所示。

例程4-29

```
第1行  <SCRIPT LANGUAGE="JavaScript">
第2行      var startNum=1;//起点数值
第3行      var stopNum=100;//终点数值
第4行      var resultNum=0;//保存计算结果
第5行      var cycleNum=startNum;//循环数
第6行
第7行      //循环数小于等于终点数值时循环
第8行      while(cycleNum<=stopNum){
第9行          resultNum+=cycleNum;//每次循环数都加上循环数
第10行         ++cycleNum;//cycyleNum递增
第11行     };
第12行     alert(resultNum);//显示结果
第13行 </SCRIPT>
```

程序设计中还会常用到限制数的范围，比如：两位数的加法就要求被加的两个数大于10小于99，其方法是利用余数方法解决，如例程4-30所示。

例程4-30

```
第1行  <Script Language="JavaScript">
第2行      //Math.random()产生0-1之间的随机数，乘以1000后变为0-1000之间的随机数，
第3行      //通过parseInt转换为0-1000之间的随机整数
第4行      //除以90，其余数为0-89,
第5行      //加上10后其范围介于10-99之间
第6行      var numFirst=parseInt(Math.random()*1000)%90+10;
第7行      var numSecond=parseInt(Math.random()*1000)%90+10;
第8行
第9行      alert(numFirst+numSecond);//仅仅显示结果
第10行     //显示出两个数以及加号、等号和结果，注意括号的应用
第11行     //删除内层括号观察效果
第12行     alert(numFirst+"+"+numSecond+"="+(numFirst+numSecond));
第13行 </Script>
```

余数用途很多，如例程4-31所示，其功能是求随机一位数的加减乘除，要求加减乘除的出现无规律。第8行是产生0-3之间的随机数，然后用0对照加法、1对照减法，以此类推，即完成对照关系。

例程4-31

```
<Script Language="JavaScript">
        //Math.random()产生0-1之间的随机数，乘以1000后变为0-1000之间的随机数，
        //通过parseInt转换为0-1000之间的随机整数
        //除以90，其余数为0-89,
        //加上10后其范围介于10-99之间
        var numFirst=parseInt(Math.random()*10);
        var numSecond=parseInt(Math.random()*10);
        var opRule=parseInt(Math.random()*100)%4;//余数为0-3，共计四种状态
        switch(opRule){
                case 0://对应为加法
                        alert(numFirst+"+"+numSecond+"="+(numFirst+numSecond));
                        break;
                case 1://对应为减法
                        alert(numFirst+"-"+numSecond+"="+(numFirst-numSecond));
                        break;
                case 2://对应为乘法
                        alert(numFirst+"×"+numSecond+"="+(numFirst*numSecond));
                        break;
                case 3://对应为除法
                        alert(numFirst+"÷"+numSecond+"="+(numFirst/numSecond));
                        break;
        }
</Script>
```

3、关系运算符

关系运算符在算术和计算机语言中都常见，如在“5>2”中大于符号就是关系运算符，用于描述两个运算数的关系，其结果为true或者false。JavaScript提供了8种关系运算符，如表3所示。

表3：JavaScript的关系运算符

运算符	描述	案例	结果
>	大于	var bResult=X>Y;	bResult结果true
>=	大于等于	var bResult=X>=Y;	bResult结果true
<	小于	var bResult=X<Y;	bResult结果false
<=	小于等于	var bResult=X<=Y;	bResult结果false
==	相等	var bResult=X==Y;	bResult结果false
===	等同	var bResult=X===Y;	bResult结果false
!=	不等	var bResult=X!=Y;	bResult结果true
!==	不等同	var bResult=X!==Y;	bResult结果false
本表中X=10，Y=5。			

关系运算符中的==与赋值运算符=容易混淆。==是判断符号左右是否相等，其结果是true或者false；而=是将符号右边的结果赋值给左边，该表达式的值是该变量的值。如例程4-32所示。

例程4-32

```
<SCRIPT LANGUAGE="JavaScript">
        var X=10,Y=5;
        alert(X==Y);//显示为false,不改变X和Y的值；
```

```
第4行            if(X=Y){//注意是=不是==
第5行                alert("X等于Y");//该语句被执行
第6行            }else{
第7行                alert("X不等于Y");//该语句不被执行
第8行            }
第9行            alert(X);//显示5
第10行       </SCRIPT>
```

第4行的if(X=Y)本意是X==Y即判断X与Y是否相等，写成X=Y却是将Y赋值给X，且表达式的值为X的值，即5，if(X=Y)就变为if(5)。当if后的括号内容不为0时，即为真(true)。由于上述原因，第9行显示为5。如果第4行被修改为if(X==Y)，则第9行显示为10。

相等(==)和等同(===)运算符也容易相混。当运算符左右两侧数据类型相同时，这两个运算符没有区别；否则，则有区别，不同时相等运算符经过转换后进行比较，如果相等则为true，否则为false，而等同运算符在数据类型不同时，则直接返回false，如例程4-33所示。

例程4-33

```
第1行       <SCRIPT LANGUAGE="JavaScript">
第2行           var X=10,Y="10";//注意：Y是字符型10.
第3行           alert(X==Y);//显示为true
第4行           alert(X===Y);//显示为false;
第5行           alert(X===parseInt(Y));//显示true
第6行       </SCRIPT>
```

4、逻辑运算符

JavaScript中提供了三种逻辑运算符，&&(and，和)、||(or、或)以及!(not，非)。当&&连接双方都为true时，结果为true，否则为false;当||连接双方一方为true时，结果为true；当!后为true时，结果为false，当!后为false时，结果为true。

例程4-34

```
第1行       <SCRIPT LANGUAGE="JavaScript">
第2行           //根据学生分数显示信息
第3行           var studentScore=78;
第4行           if(studentScore>=85)alert("优秀!");
第5行           if(studentScore>=75 && studentScore<85)alert("良好!");//大于等于75且小于85
第6行
第7行           if(studentScore<60 || studentScore>=85)alert("特别!");//小于60或者大于等于85
第8行           if(studentScore<60)alert("不及格!");//小于60不及格
第9行           if(!(studentScore<60))alert("及格!");//非小于60
第10行      </SCRIPT>
```

“0<N<5”在数学中表示N大于0且小于5，在JavaScript中，不能如此表达，正确表达形式是“(N>0 && N<5)”，即将此分解为两个表达式并用逻辑运算符连接。数学中的很多取值范围表达，都需做类似的转换。

5、其他运算符

除了上述运算符外，还有正数运算符(+)、负数运算符(-)、字符连接运算符(+)、条件运算符(?)、逗号运算符(,)等。正数和负数运算符置于数值前，分别表示整数和负数，正数运算符常省略，如“X=+5;Y=-5;”等。

⑴ 字符连接运算符(+)

加号(+)除用于数值相加、正数符号外，还用于字符串连接，例程4-35是+的几种应用场景。

例程4-35

```
第1行       <SCRIPT LANGUAGE="JavaScript">
第2行           var firstName="张";
第3行           var givenName="珊";
```

```
第4行            var fullName=firstName+givenName;
第5行            alert(fullName);//显示张珊
第6行
第7行            var X1=100,X2="99";
第8行            alert(X1+X2);//显示10099
第9行            alert(X1-X2);//显示1
第10行           alert(X1-(-X2));//显示199
第11行           alert(X1+parseInt(X2));//显示199
第12行         </SCRIPT>
```

例程4-35中，第4行代码将两个字符串拼接在一起并将结果赋值给fullName。第8行是数值型数据与字符型数据做+运算。在这种情况下，默认将数值型数据转换成字符型数据然后运算，所以结果为10099。第9行虽然也是数值型数据和字符型数据混合运算，但由于其他数据类型不能做-运算，因此JavaScript将把数据转换为数值型，然后进行运算，所以结果为1。与此类似的是第10行。

⑵ 条件运算符(?)

条件运算符类似if语句，其格式如下，例程4-36是其应用。当条件表达式值为true时，表达式结果为trueValue，否则为falseValue。

条件表达式?trueValue:falseValue

例程4-36

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        var Num1=200,Num2=100;
第3行
第4行        //如果Num1大于Num2，则表达式的值为Num1,
第5行        //否则表达式的值为Num2，调整Num1和Num2
第6行        //maxNum总为其中最大一个数
第7行        var maxNum=Num1>Num2?Num1:Num2;
第8行
第9行        alert(maxNum);
第10行
第11行       //与第18行效果相似，但增加了代码量
第12行       if(Num1>Num2){
第13行           alert(Num1);
第14行       }else{
第15行           alert(Num2);
第16行       }
第17行   </SCRIPT>
```

在例程4-36中，第7行代码的含义是当Num1>Num2时，则maxNum等于Num1的值，否则等于Num2的值。如果采用if语句实现，则需要更多的代码，且执行效率有所下降。

⑶ 逗号运算符(,)

逗号运算符可以实现在一条语句中执行多个运算，常用于变量申明中。当一个语句中有多个运算时，以最后一个运算作为该语句或表达式的值，如例程4-37所示。

例程4-37

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        var Num1=100,Num2=99;
```

```
第3行          if(Num1=100,Num2=0){//条件表达式的值为Num2，为0，即false
第4行                  alert(Num2);
第5行          }else{
第6行                  alert(Num1);//执行该语句
第7行          }
第8行    </SCRIPT>
```

第六节　函数

计算机语言中的函数与数学中的函数有些类似。数学函数y=f(x)中，x称为自变量，y称为因变量，或者称为函数的值，f是function的简写，称之为函数，f可以表示各种类型的函数，正弦函数，二次函数等。在计算机语言中，x称为参数，参数可以有一个，也可以有多个。f表现为各种函数名称，如alert("程序设计 思维体操")中，alert相当于函数名称，其括号中内容是参数。在计算机语言中，有语言提供的最常用的系统函数(内部函数)，也支持自定义函数。

1、自定义函数

例程4-38第3行到第6行自定义一个名为sumNum()的函数，该函数有两个参数a和b，在其后的花括弧中，计算出a和b的值，并用return返回该值。如果没有return语句，则该函数没有返回值，这在很多计算机语言中是允许的。

例程4-38

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行          //此处开始定义sumNum函数
第3行          function sumNum(a,b){
第4行                  var sum=a+b;
第5行                  return sum;//函数用return返回函数的结果
第6行          }
第7行
第8行          var firstNum=100;
第9行          var secondNum=99.12;
第10行
第11行         //下一行调用定义的函数，变量sumResult记录函数结果
第12行         var sumResult=sumNum(firstNum,secondNum);
第13行         alert(sumResult);//显示函数执行结果
第14行   </SCRIPT>
```

在例程4-38中，第12行调用sumNum()函数，输入参数firstNum和secondNum两个参数的值，并将结果保存在sumResult变量中。虽然定义函数时使用的a和b，调用函数时则没有必要用a和b输入参数值，甚至可以直接输入参数如sumNum(10,100)亦可。

在自定义函数中，当执行语句遇到return语句时，将退出函数并返回函数值。注意：如果return后没有函数的返回值，也将直接退出函数，此时结果默认为undefined。

例程4-39是求3-100之间的质数，在其第7-15行定义一个名为isPrime()函数，用于判断一个数是否质数。判断方法是如果程序中的checkNum能整除inputNum，则表明不是质数，则return false，表示输入的inputNum不是质数。程序中checkNum从2开始，一直增长到inputNum-1为止，如果都没有发现能被整除的数，则表明是质数，此时return true;(第14行)。注：checkNum没有必要一定到inputNum-1，到其二分之一即可，甚至可以更小。还可以通过多种方法减少循环次数，减小计算机计算量。

例程4-39

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行          var startNum=3;
第3行          while(startNum<=100){//小于100循环，即100以内的质数
第4行                  if(isPrime(startNum)==true)alert(startNum+"是一个质数！");
第5行                  startNum++;
第6行          }
```

```
第7行            function isPrime(inputNum){
第8行                var checkNum=2;
第9行                while(true){
第10行                    if(inputNum%checkNum==0)return false;//如果被checkNum整除，即不为整数
第11行                    checkNum++;
第12行                    if(checkNum>=inputNum)break;//当checkNum大于等于inputNum时，退出循环
第13行                }
第14行                return true;//上述所有数都不能整除，才会被执行，此时为质数
第15行            }
第16行    </SCRIPT>
```

例程4-40是对例程4-4的改造，其中的changeImg()是一个没有返回值的函数，甚至没有参数的函数。在调用函数时，没有参数的函数，其函数名后的括号也不能省略。

例程4-40

```
第1行    <HTML>
第2行        <HEAD>
第3行            <TITLE>单击图片切换</TITLE>
第4行            <Script type="Text/JavaScript" language="JavaScript" src="./jquery-1.4.2.min.js"></Script
    >
第5行            <META http-equiv="Content-type" content="text/html; charset=gb2312">
第6行        </HEAD>
第7行
第8行        <BODY>
第9行            <IMG src="http://www.pku.edu.cn/images/index/main_img/1.gif" style="width:200px" ImgNo
    ="1" ID=myImg>
第10行           <SCRIPT LANGUAGE="JavaScript">
第11行               //单击图片切换图片
第12行               $("#myImg").click(function(){
第13行                   changeImg();
第14行               });
第15行
第16行               //切换图片函数，该函数没有参数，没有返回值
第17行               function changeImg(){
第18行                   var webSite="http://www.pku.edu.cn/images/index/main_img/";
第19行                   var currentImgNo=$("#myImg").attr("ImgNo");//取出当前图片编号
第20行                   var nextImgNo=parseInt(currentImgNo)+1;//计算出下一个图片编号
第21行
第22行                   //nextImgNo大于10则让nextImgNo等于1，相当于从头开始
第23行                   if(nextImgNo>10){
第24行                       nextImgNo=1;
第25行                   }
第26行
第27行                   var imgSRC=webSite+nextImgNo.toString()+".gif";//合成IMG标志的src属性
第28行
第29行                   $("#myImg").attr("src",imgSRC);//改变click图片的src属性
```

```
                $("#myImg").attr("ImgNo",nextImgNo);//修改ImgNo属性，保存当前编号
            }
        </SCRIPT>
    </BODY>
</HTML>
```

例程4-41英文数字到汉字数字的转换，从中可以看出一个函数定义后能多次重复引用，实现“代码复用”，这是函数的重要作用。

例程4-41

```
<Script Language="JavaScript">
    var chnNum=convertEng2Chn(5);
    alert(chnNum);
    //将一个数转换为汉字模式
    var num=8964;
    var temp=num;
    var result="";//用于保存转换结果

    while(temp>0){//用temp循环，不改变num的值，当temp等于0时停止循环
        var unitNum=temp%10;//除以10的余数是个位数字
        result=convertEng2Chn(unitNum)+result;//将个位转换后的结果与先前结果相加
        temp=(temp-temp%10)/10;//减去个位数以后除以10，得到没有个位数字的数
    }

    alert(result);//显示捌玖陆肆

    function convertEng2Chn(Num){
        var Chn=["零","壹","贰","叁","肆","伍","陆","柒","捌","玖"];
        return Chn[Num];
    }
</Script>
```

2、系统函数

⑴ alert()：消息对话框

alert()函数用于显示参数的内容，如果参数不是字符串将自动转换为字符串，如例程4-42。

例程4-42

```
<Script Language="JavaScript">
    var Num=123;
    alert(Num);//显示为123
    alert("XYZ");//显示为XYZ
    alert(Num+"XYZ");//显示为123XYZ
    alert(123==Num);//显示为true
    alert(123==Num+"转换为字符串");//显示为false，注意：先运算+，然后运算==。
</Script>
```

灵活运用alert()函数，有助于程序调试。在需要观察代码执行效果时，可以插入alert()函数及其相关参数，如例程4-43。

例程4-43

```
<Script Language="JavaScript">
```

```
第2行          var chnNum=convertEng2Chn(5);
第3行          alert(chnNum);
第4行          //将一个数转换为汉字模式
第5行          var num=8964;
第6行          var temp=num;
第7行          var result="";//用于保存转换结果
第8行          while(temp>0){//用temp循环，不改变num的值，当temp等于0时停止循环
第9行                  var unitNum=temp%10;//除以10的余数是个位数字
第10行                 result=convertEng2Chn(unitNum)+result;//将个位转换后的结果与先前结果相加
第11行                 temp=(temp-temp%10)/10;//减去个位数以后除以10，得到没有个位数字的数
第12行                 alert(temp);//每次循环都能观察到temp值的变化，有利调试
第13行         }
第14行         alert(result);//显示捌玖陆肆
第15行         function convertEng2Chn(Num){
第16行                 var Chn=["零","壹","贰","叁","肆","伍","陆","柒","捌","玖"];
第17行                 return Chn[Num];
第18行         }
第19行   </Script>
```

例程4-43与例程4-41相比，在第11行后增加了“alert(temp);”，每次循环都可以显示出temp的变化，以利于观察循环过程。从这个意义上讲，alert()相当于输出执行过程中的结果，以利于观察。

⑵ confirm()：确认对话框

图4-10是confirm()执行的界面。confirm()函数中，提示信息以字符串的形式放在括号中。如例程4-44所示。该函数执行的返回值为true(选择确定)或者false(选择取消)。

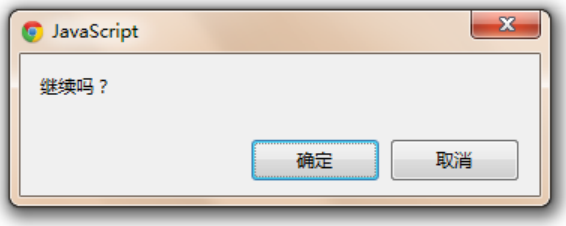

图4-10 confirm()函数执行效果图

例程4-44

```
第1行   <SCRIPT LANGUAGE="JavaScript">
第2行          while(true){
第3行                  var returnVal=confirm("继续吗？");
第4行                  if(returnVal==false){
第5行                          break;
第6行                  }
第7行          }
第8行   </SCRIPT>
```

在例程4-44中，while(true)表示如果没有中断将无限循环。第3行通过cofirm()函数询问是否继续，并选择结果保存到变量returnVal中，如果选择“确定”，returnVal为true，如果选择returnVal等于false。第4到第6行对返回值为false进行处理，即选择“取消”终止循环。

⑶ prompt()：输入对话框

prompt()用于显示提示用户进行输入的对话框，格式为prompt(text,defaultText)，其中text为提示信息，defaultText为默认输入内容。如例程4-45所示。注意：如果选择取消，则返回值为null。

例程4-45

```
<SCRIPT LANGUAGE="JavaScript">
        //判断输入数值奇偶性
        var inputVal=prompt("请输入整数",100);
        if(inputVal%2==0){
                alert("这是一个偶数！");
        }else{
                alert("这是一个奇数！");
        }
</SCRIPT>
```

(4) setInterval()/clearInterval()：设置/取消周期执行函数

setInterval()能周期执行指定代码，其格式为setInterval("被执行代码",间隔时间)，参见例程4-46。注意：被执行代码为字符串。

例程4-46

```
<HTML>
    <HEAD>
        <TITLE>单击图片切换</TITLE>
        <Script type="Text/JavaScript" language="JavaScript" src="./jquery-1.4.2.min.js"></Script>
        <META http-equiv="Content-type" content="text/html; charset=gb2312">
    </HEAD>

    <BODY>
        <IMG src="http://www.pku.edu.cn/images/index/main_img/1.gif" style="width:200px" ImgNo="1" ID=myImg>
        <SCRIPT LANGUAGE="JavaScript">
            //实现定时切换，单击图片切换
            $("#myImg").click(function(){
                changeImg();
            });

            //定时切换，
            setInterval("changeImg()",5000);//5000毫秒，可根据需要调整该数值

            //切换图片函数
            function changeImg(){
                var webSite="http://www.pku.edu.cn/images/index/main_img/";
                var currentImgNo=$("#myImg").attr("ImgNo");//取出当前图片编号
                var nextImgNo=parseInt(currentImgNo)+1;//计算出下一个图片编号

                //nextImgNo大于10则让nextImgNo等于1，相当于从头开始
                if(nextImgNo>10){
                    nextImgNo=1;
                }
```

```
第29行
第30行                                    var imgSRC=webSite+nextImgNo.toString()+".gif";//合成IMG标志的src属性
第31行
第32行                                    $("#myImg").attr("src",imgSRC);//改变click图片的src属性
第33行                                    $("#myImg").attr("ImgNo",nextImgNo);//修改ImgNo属性，保存当前编号
第34行                                }
第35行                            </SCRIPT>
第36行                    </BODY>
第37行            </HTML>
```

相对于前面的例程，增加了第16、17行，其含义是每隔5000秒执行changeImg()函数。该程序的功能在很多网页中都有使用。网页的空间有限，如何在有限的空间呈现更多信息总是一个挑战。该例程实现了在一个图片空间，通过自动或者单击鼠标方式呈现更多图片。

根据需求，有时还需取消定时行为，其方式是通过clearInterval()函数，其使用方式如例程4-47所示。

例程4-47

```
第1行    //定时切换，
第2行    var myTimer=setInterval("changeImg()",5000);//5000毫秒，可根据需要调整该数值
第3行    $("#myImg").dblclick(function(){
第4行            clearInterval(myTimer);
第5行    });
```

和例程4-46相比，变量myTimer等于setInterval()函数的返回值，且在$("#myImg")的双击事件中，通过clearInterval(myTimer)取消。通过该代码，将在双击图片，取消定时切换图片功能。setInerval()和clearInterval()通过共同的变量联系在一起，一个负责设置启动，一个负责取消。

(5) setTimeout()/clearTimeout()：设置/取消延迟执行函数

setTimeout()和clearTimeout()函数与setInterval()和clearInterval()函数功能相似，只是setTimeout(“被执行代码”,间隔时间)到设定的间隔时间后仅仅执行一次就不再执行，而setInterval()则重复执行。另外，可用setTimeout()函数模拟setInterval()。如例程4-48所示，在其第2行，到5000毫秒时执行一次timeChangeImg()，而在该函数中，执行changeImg()，且同时又在此启动timeChangeImg()，依次类推，实现重复执行，达到setInterval()效果。

例程4-48

```
第1行    //定时切换，
第2行    setTimeout("timeChangeImg()",5000);//5000毫秒，可根据需要调整该数值
第3行    function timeChangeImg(){
第4行            changeImg();
第5行            setTimeout("timeChangeImg()",5000);
第6行    }
```

第七节　常用对象

面向对象是现代高级语言的重要特征，如C++、Java、C#等，JavaScript也具有面向对象(Oriented Object，简称OO)特征。对象是具有属性(property)和方法(Method)复杂数据，属性用于刻画对象特征，方法用于描述对象的数据处理能力，是集成在对象中的函数。对象与属性和方法之间用点运算符(.)连接，Math是JavaScript提供内置对象，Math.PI是Math对象PI属性，Math.random()是Math对象的random()方法。

JavaScript已内置常用对象，如：Math(数学)、Date(日期)、Array(数组)、String(字符串)、RegExp(正则表达式)等，以及与浏览器相关对象等，如Screen、Location、History、Navigator、PopupAlert、Timing、Cookies等。本章讲到的alert实际就是PopupAlert对象，可以写成window.alert(显示内容表达式)。在JavaScript中，window对象是系统默认，可以省略，因此常写成alert()，而很少写成widow.alert。Timing与之类似，即为本章讲到的4个时钟函数(setInterval/clearInterval、setTimeout/clearTimeout)。

Math、Date、Array、String将在本章讲到，其他对象将在相关章节讲解。

1、Math对象

Math对象主要用于处理与数学相关的内容，主要集成一些常数及常用数学处理，如三角函数、随机函数等，如表4所示。

表4：Math对象的属性和方法

Math对象的属性			
属性	描述	示例	结果
E	返回算术常量 e，即自然对数的底数（约等于2.718）。	var X=Math.E	2.718281828459045
LN2	返回 2 的自然对数（约等于0.693）。	var X=Math.LN2	0.6931471805599453
LN10	返回 10 的自然对数（约等于2.302）。	var X=Math.LN10	2.302585092994046
LOG2E	返回以 2 为底的 e 的对数（约等于 1.414）。	var X=Math.LOG2E	1.4426950408889633
LOG10E	返回以 10 为底的 e 的对数（约等于0.434）。	var X=Math.LOG10E	0.4342944819032518
PI	返回圆周率（约等于3.14159）。	var X=Math.PI	3.141592653589793
SQRT1_2	返回1/2的平方根(约等于 0.707)。	var X=Math.SQRT1_2;	0.7071067811865476
SQRT2	返回 2 的平方根（约等于 1.414）。	var X=Math.SQRT2;	1.4142135623730951
Math对象的方法			
acos(x)	反余弦值。		
asin(x)	反正弦值。		
atan(x)	反正切值(x介于-PI/2与PI/2弧度之间)。		
atan2(y,x)	x轴到点(x,y)的角度(介于-PI/2与PI/2弧度之间)。		
cos(x)	返回数的余弦。		
sin(x)	返回数的正弦。		
tan(x)	返回角的正切。		
abs(x)	绝对值。	var X=Math.abs(-101);	101
ceil(x)	对数进行上舍入。	var X=Math.ceil(-10.09);	-10
floor(x)	对数进行下舍入。	var X=Math.floor(-10.09);	-11
round(x)	把数四舍五入为最接近的整数。	var X=Math.round(10.59);	11
exp(x)	返回 e 的指数。	var X=Math.exp(2);	7.38905609893065
log(x)	自然对数(底为e)。	var X=Math.log(10);	2.302585092994046
max(x1,x2,……,xn)	最大值	var X=Math.max(1,2,3,4,7);	7
min(x1,x2,……,xn)	最小值	var X=Math.min(1,2,3,4,7);	1
pow(x,y)	x的y次幂	var X=Math.pow(2,3);	8
random()	返回0~1之间的随机数。	var X=Math.random();	随机变化
sqrt(x)	返回数的平方根。	var X=Math.sqrt(8);	2.8284271247461903

某些网站首页的形象图片随机出现，可采用Math.random()实现，如例程4-49(假设有文件名为1.gif，2.gif，……，10.gif共计10幅图片)。

例程4-49

```
第1行  <HTML>
第2行      <HEAD>
第3行          <TITLE>随机图片</TITLE>
第4行          <SCRIPT LANGUAGE="JavaScript" src="jquery-1.9.1.min.js"></SCRIPT>
第5行      </HEAD>
第6行
```

```
第7行          <BODY>
第8行          <Div ID="imgBox"></Div>
第9行              <SCRIPT LANGUAGE="JavaScript">
第10行
第11行                  //Math.random()*100得到0-100之间的实数
第12行                  //parseInt(Math.random()*100)为0-100之间整数
第13行                  //parseInt(Math.random()*100)%10得到0-9之间的整数
第14行                  //编号从1开始，因此加上1
第15行
第16行                  var imgNo=parseInt(Math.random()*100)%10+1;//图片编号
第17行                  var imgName=imgNo+".gif";//根据编号形成图片名称
第18行                  $("#imgBox").html("<img src='"+imgName+"'>");//图片相关的HTML代码
第19行              </SCRIPT>
第20行          </BODY>
第21行      </HTML>
```

2、Date对象

Date对象有广泛的用途。比如：日历应用，Web页面上日期选择等。在使用日期对象前，须初始化，其格式如下：

```
new Date();//计算机当前日期和时间
new Date(milliseconds);//从1970年1月1日计起的毫秒数
new Date(datestring);//Datestring为字符串，格式为月-日-年 时:分:秒
new Date(year, month);//注意：一月为0，二月为1，以此类推
new Date(year, month, day)
new Date(year, month, day, hours)
new Date(year, month, day, hours, minutes)
new Date(year, month, day, hours, minutes, seconds)
new Date(year, month, day, hours, minutes, seconds, microseconds)
```

表5是Date对象相关的方法，其中星期的编号从0开始，即星期天为0，星期一为1；月份编号也是从开始，即一月为0，十二月为11；时钟秒分别介于0~59；毫秒介于0~999。

表5：Math对象的属性和方法

方法	描述	示例	结果
Date()	日期对象初始化。	var myDate=new Date(2013,0,1,9,48,58,100);	初始化为指定时间
getDate()	从Date对象返回一个月中的某一天 (1 ~ 31)。	alert(myDate.getDate());	1
getDay()	从Date对象返回一周中的星期 (0 ~ 6)。	alert(myDate.getDate());	2
getMonth()	从Date对象返回月份 (0 ~ 11)。	alert(myDate.getMonth());	0
getFullYear()	从Date对象返回四位数字年份。	alert(myDate.getFullYear());	2013
getHours()	从Date对象返回小时(0 ~ 23)。	alert(myDate.getHours());	9
getMinutes()	从Date对象返回分钟(0~59)。	alert(myDate.getMinutes());	48
getSeconds()	从Date对象返回秒数(0~59)。	alert(myDate.getSeconds());	58
getMilliseconds()	从Date对象返回毫秒数(0~999)。	alert(myDate.getMilliseconds());	100
getTime()	返回1970年1月1日至Date设定时间的毫秒数。	alert(myDate.getTime());	1357004938100
getTimezoneOffset()	返回本地时间与格林威治标准时间(GMT)的分	alert(myDate.getTimezoneOffset());	显示-480

	钟差。		
getUTCDate()	根据世界时从Date对象返回月中的一天 (1 ~ 31)。	alert(myDate.getUTCDate());	1
getUTCDay()	根据世界时从Date对象返回周中的一天 (0 ~ 6)。	alert(myDate.getUTCDay());	2
getUTCMonth()	根据世界时从 Date 对象返回月份 (0 ~ 11)。	alert(myDate.getUTCMonth());	0
getUTCFullYear()	根据世界时从 Date 对象返回四位数的年份。	alert(myDate.getUTCFullYear());	2013
getUTCHours()	根据世界时返回 Date 对象的小时 (0 ~ 23)。	alert(myDate.getUTCHours());	1
getUTCMinutes()	根据世界时返回 Date 对象的分钟 (0 ~ 59)。	alert(myDate.getUTCMinutes());	48
getUTCSeconds()	根据世界时返回 Date 对象的秒钟 (0 ~ 59)。	alert(myDate.getUTCSeconds());	58
getUTCMilliseconds()	根据世界时返回 Date 对象的毫秒(0 ~ 999)。	alert(myDate.getUTCMilliseconds());	100
parse()	从1970年1月1日午夜到指定日期(字符串)的毫秒数。	alert(Date.parse("1-1-2013"));	1356969600000
setDate()	设置 Date 对象中月的某一天 (1 ~ 31)。	myDate.setDate(20);	设置为20号
setMonth()	设置 Date 对象中月份 (0 ~ 11)。	myDate.setMonth(11);	设置为12月
setFullYear()	设置 Date 对象中的年份（四位数字）。	myDate.setFullYear(2014);	设置为2014年
setHours()	设置 Date 对象中的小时 (0 ~ 23)。	myDate.setHours(0);	设置为0点
setMinutes()	设置 Date 对象中的分钟 (0 ~ 59)。	myDate.setMinutes(57);	设置为57分
setSeconds()	设置 Date 对象中的秒钟 (0 ~ 59)。	myDate.setSeconds(56);	设置为56秒
setMilliseconds()	设置 Date 对象中的毫秒 (0 ~ 999)。	myDate.setMilliseconds(123);	设置为123毫秒
setTime()	与1970年1月1日夜间偏离的毫秒数	myDate.setTime(1234);	过了1秒234毫秒
setUTCDate()	根据世界时设置 Date 对象中月份的一天 (1 ~ 31)。	myDate.setUTCDate(1);	
setUTCMonth()	根据世界时设置 Date 对象中的月份 (0 ~ 11)。	myDate.setUTCMonth(1);	
setUTCFullYear()	根据世界时设置 Date 对象中的年份（四位数字）。	myDate.setUTCFullYear(2013);	
setUTCHours()	根据世界时设置 Date 对象中的小时 (0 ~ 23)。	myDate.setUTCHours(1);	
setUTCMinutes()	根据世界时设置 Date 对象中的分钟 (0 ~ 59)。	myDate.setUTCMinutes(56);	
setUTCSeconds()	根据世界时设置 Date 对象中的秒钟 (0 ~ 59)。	myDate.setUTCSeconds(56);	
setUTCMilliseconds()	根据世界时设置 Date 对象中的毫秒 (0 ~ 999)。	myDate.setUTCMilliseconds(123);	
toString()	把 Date 对象转换为字符串。	alert(myDate.toString());	显示为Tue Jan 1 09:48:58 UTC+0800 2013
toTimeString()	把 Date 对象的时间部分转换为字符串。	alert(myDate.toTimeString());	显示为09:48:58 UTC+0800

toDateString()	把 Date 对象的日期部分转换为字符串。	alert(myDate.toDateString());	显示为Tue Jan 1 2013
toGMTString()	请使用 toUTCString() 方法代替。	alert(myDate.toGMTString());	显示为Tue, 1 Jan 2013 01:48:58 UTC
toUTCString()	根据世界时，把 Date 对象转换为字符串。	alert(myDate.toUTCString());	显示为Tue, 1 Jan 2013 01:48:58 UTC
toLocaleString()	根据本地时间格式，把 Date 对象转换为字符串。	alert(myDate.toLocaleString());	显示为2013年1月1日 9:48:58
toLocaleTimeString()	根据本地时间格式，把 Date 对象的时间部分转换为字符串。	alert(myDate.toLocaleTimeString());	显示为9:48:58
toLocaleDateString()	根据本地时间格式，把 Date 对象的日期部分转换为字符串。	alert(myDate.toLocaleDateString());	显示为2013年1月1日
本例中myDate为2013年1月1日上午9:48:58.100			

例程4-50是一个有关Date的简单应用。

例程4-50

```
<SCRIPT LANGUAGE="JavaScript">
        //当月第一天星期几?
        var currentDate=new Date();
        var firstDayWeek=new Date(currentDate.getFullYear(),currentDate.getMonth(),1).getDay();

        //二月最后一天是几号?
        //注意是三月第0天，实际上为2月最后一天，
        //new Date(2013,2,-1)则是二月倒数第二天
        var lastDayOfFeb=new Date(2013,2,0);
        alert(lastDayOfFeb.toLocaleString());//2013年2月28日
</SCRIPT>
```

3、数组对象(Array)

所谓数组，就是多数据按顺序排列在一个变量名称中，且可用变量名称及其索引值读写(R&W)数组成员，类似数学中的X1,X2,……,Xn，X表示变量名称，1,2,……,n下标表示索引值。如var myArr=[2,5,6,10,20];在JavaScript中可以表示申明了一个名为myArr的数组，2,5,6,10,20是保存在myArr中的数据，实现了一个变量名称保存多个数据，可以用myArr[0]访问第一个成员2，编号为0，myArr[4]访问第5个成员，其值为20。也可以通过myArr[3]=120改变编号为3的成员值，原值10改变为120。注意：和很多编程语言一样，JavaScript数组索引编号从0开始。

⑴ Array对象申明

数组申明方式如下。在方式1中，申明一个空的数组；方式2则申明一个能包含多少个数据的数组；方式3则在申明数组的同时，同时包含了数据成员。方式4、方式5是两种申明数组的简明方式。

Array对象申明方式

方式1：var arrName=new Array();

方式2：var arrName=new Array(size);

方式3：var arrName=new Array(element1,element2,……,elementN);

方式4：var arrName=[];

方式5：var arrName=[element1,element2,……,elementN];

备注：arrName是数组名称，与变量命名规则相同。

通过下标可以存取数组成员，如例程4-51所示，其第5行相当于向数组成员赋值，即相当于dataList[0]等于多少，dataList[1]等于多少，而第11行相当于分别依次显示dataList[0]，dataList[1]……。

例程4-51

```
<SCRIPT LANGUAGE="JavaScript">
        var dataList=new Array();//也可以写成var dataList=[];
        var count=0;
        while(count<10){
                dataList[count]=parseInt(Math.random()*100);
                count++;
        }//本循环依次向dataList[0],dataList[1],……,dataList[9]存入数值

        count=0;//删除本句将不显示，因count=10，不满足循环条件
        while(count<10){
                alert(dataList[count]);
                count++;
        }//本循环依次显示dataList[0],dataList[1],……,dataList[9]中的数值
</SCRIPT>
```

JavaScript的数组和其他语言不尽相同。Javascript的数组在申明时可以不指定数组成员的数量，然后根据需要增加数组长度(增加成员)，或者减少数组长度(减少成员)。另外，向数组存入数据时，不必按编号顺序写入，如例程4-52所示。在其他很多语言中，申明数组时必须指明数组长度，在运行过程中不允许增减数据成员数量，如C语言等。JavaScript的数组和C++的vector相似，但是JavaScript的数组可以存入不同类型的数据(不是好习惯)，而C++的vector是具有JavaScript类似的能力。

例程4-52

```
<SCRIPT LANGUAGE="JavaScript">
        var arrData=[];
        arrData[0]=100;
        arrData[100]=200;
        //arrData[1]到arrData[99]之间的数据值都为undefined(未定义)
</SCRIPT>
```

注意，JavaScript数组的编号从0开始。

(2) Array对象属性和方法

为更好地操作数组，JavaScript提供了一个常用的length(长度)属性和不少方法。例程4-53是有关length属性的使用。

例程4-53

```
<SCRIPT LANGUAGE="JavaScript">
        var arrA=[7,8,9,4,5,6,1,2,3];//初始化数组为9个成员
        alert(arrA.length);//显示arrA的长度，即arrA元素的个数，值为9;

        arrA.length=20;
        alert(arrA.length);//数组长度增大，增加了11个成员，新增成员值为undefined

        arrA.length=2;
        alert(arrA.length);//数组长度缩小，裁剪掉尾部7个成员。
</SCRIPT>
```

从例程4-53可以看出，Array.length不仅可以反应数组现有成员的数量(第3行)，还可以增加成员数量(第5行)和减少成员数量(第8行)。当增大成员数量时，新增成员的值为undefined。当减少成员数量时，仅保留从头开始指定数量的成员，尾部成员全部删除，导致数据丢失，操作要慎重。

数组的操作较多，涉及到数组合并(Array.concat)、成员增加(Array.push、Array.unshift、Array.splice)、删除(Array.pop、Array.shift、Array.splice)、截取(Array.slice)、替换(Array.splice)、排序(Array.reverse、Array.sort)以及转成成字符串

(Array.join、Array.toString、Array.toLocaleString、Array.valueOf)等。

表6：Array对象方法

方法	描述	用例
concat()	连接两个或更多的数组，并返回结果。	var temp=arrA.concat(arrB); //temp为Array，包含arrA和arrB，arrA和arrB不变化。
join()	把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔。	var temp=arrA.join("#"); //arrA不发生变化，temp为String，值为1#2#3。
pop()	删除并返回数组的最后一个元素	var temp=arrA.pop(); //arrA.length变为2，成员变为1,2。temp值为3。
push()	向数组的末尾添加一个元素，并返回新的长度。	var temp=arrA.push(1234); //arrA.length变为4，temp值为4即arrA.length值。
shift()	删除并返回数组的第一个元素	var temp=arrA.shift(); //arrA.length变为2，值序为2,3。temp值为arrA中被删除元素。
unshift()	向数组的开头添加一个或更多元素。格式为：unshift(element1,element2,……)	var temp=arrA.unshift(100,200,300,400); //向数组开始增加新元素，第1个元素编号为0，一次类推，arrA的值序为100,200,300,400,1,2,3。temp值为arrA的数组长度，在IE浏览器上temp值为undefined。
slice()	从数组返回选定的部分元素。该方法有两个参数，第1个为必须，第2个可选。	var temp=arrB.slice(2,5); //arrB不发生变化，temp为Array拥有6,7,8。如果省略slice第二个参数，则从第1个参数开始到最后一个元素。如果slice参数为负数，则表示从尾部开始计数，最后一个元素编号为-1，如var temp=arrB.slice(-5,-3);则temp为有6,7两个元素的Array。

splice()	插入、删除或替换数组的元素。格式为splice(index,howmany,element1,element2,……)，index为元素起点编号，howMany为从index开始删除的个数，如果是0则不删除，element1、element2等为插入的元素。	var temp=arrB.splice(2,3);
		//删除从编号为2的元素开始的3个元素，即6,7,8，temp为Array，元素为被删除的元素，即6,7,8。arrB为4,5,9,10。
		var temp=arrB.splice(2,3,100,200,300,400);
		//删除从编号2开始的3个元素，即6,7,8。同时从编号2开始插入4个元素，arrB值序为4,5,100,200,300,400,9,10。temp为Array，结果为6,7,8。

reverse()	颠倒数组中元素的顺序。	var temp=arrA.reverse(); //arrA的值序变为3,2,1,temp与arrA相等。
sort()	对数组的元素进行排序。该方法可有1个排序函数作参数，指定排序规则。	var temp=arrB.sort(); //arrB被排序后值序为10,4,5,6,7,8,9。sort()如没有参数，则将数组转换为字符串后按字母排序。排序函数参见例程5-3。
toString()	把数组转换为字符串，并返回结果。	var temp=arrA.toString(); //temp为String，结果为1,2,3，每个数据之间用逗号分开。
toLocaleString()	把数组转换为本地数组，并返回结果。	var temp=arrA.toLocaleString(); //temp为String，值为1.00，2.00，3.00。每个值之间用中文逗号分开。
valueOf()	返回数组对象的原始值	var temp=arrA.valueOf(); //temp为String，值为1,2,3，与toString()相同。
注：假设var arrA=[1,2,3],arrB=[4,5,6,7,8,9,10];//arrA.length等于3，arrB.length等于7。各用例之间互相独立。		

对数组排序是一个经常性需求，但是JavaScript默认的排序规则是将数组转换为字符串，然后按照字母顺序排序，所以就出现10,4,5,6,7,8,9这样的现象，因为10的首数字的1，其ASCII在4之前。可以通过排序函数改变JavaScript默认排序规则，如例程4-54所示。

例程4-54

```
第1行  <SCRIPT LANGUAGE="JavaScript">
第2行      var arrA=[123,23,546,879,11,34,321];
```

```
第3行
第4行          arrA.sort(sortA);//按照sortA标准排序
第5行          alert(arrA);//显示为11,23,34,123,321,546,879
第6行
第7行          arrA.sort(sortB);
第8行          alert(arrA);//显示为879,546,321,123,34,23,11
第9行
第10行         arrA.sort(sortRND);
第11行         alert(arrA);//每次显示结果不一致
第12行
第13行         //排序函数需要有两个参数
第14行         function sortA(a,b){
第15行                 return (a-b);//如果a-b大于0，小数在前
第16行         }
第17行         function sortB(a,b){
第18行                 return -(a-b);//与sortA相反，大数在前
第19行         }
第20行
第21行         function sortRND(a,b){
第22行                 return Math.random()-0.5
第23行                 /*Math.random()的取值范围0-1之间，减去0.5后，
第24行                 则介于-0.5-0.5之间，随机变化，所以将出现乱序*/
第25行         }
第26行   </SCRIPT>
```

排序函数需要有两个参数，排序函数根据返回值决定是否前后交换，若返回值小于0，则被比较的两个数前后交换，等于0或者大于0则不交换，因此只要控制返回值则控制排序标准。如果排序函数返回值随机，则会乱序。例程4-54提供了正序、反序、乱序三种排序函数。

⑶ jQuery数组操作方法

作为最常用的JavaScript库，jQuery提供不少数组操作方法，包括数组查找($.inArray)、数组合并($.merge)、数组去重($.unique)、数组遍历($.each)、数组过滤($.grep)、数组映射($.map)等，这些方法都是数组操作中较为常用的需求。

表7：jQuery数组操作方法

方法	描述	用例
$.inArray(value,array)	查找value是否在array中，返回值为元素位置或-1(该元素不存在于数组)。	var temp=$.inArray(7,arrB); //temp值为3，查找7在arrB中的位置。
$.merge(arrA,arrB)	将arrB追加到arrA之后，返回值为arrA的length值。	var temp=$.merge(arrA,arrB); //arrB追加到arrA之后，arrA长度为10，temp为arrA的长度。

$.unique(array)	让array中元素唯一，即重复元素仅留下一份。返回值为删重后的数组。	arrA.push(2);//arrA编程1,2,3,2
		var temp=$.unique(arrA);
		//删除重复的2，arrA变成1,2,3。temp与arrA同。

$.each(array,callback)	将array中每个元素用callback函数处理。	var temp=$.each(arrA,function(i,n){
		arr[i]=arr[i]*n;
		});//arrA编程1,4,9，temp与arrA相同。
		//i为数组编号，n为编号对应值，均可省略。

	数组过滤，callback函数返回值为true或false。invert部分可选值为true或false，	var temp=$.grep(arrB,function(n,i){ return n>7;

$.grep(array,callback,[invert])	可以省略。当省略时，callback函数返回值为true，则保留该元素，否则删除该元素。当invert为false，则相反。	}); //temp为Array，arrB中大于7的元素赋值到temp，值为8,9,10。

$.map(array,callback)	数组映射，即将array中每个元素用callback处理，并将处理结果返回到一个数组中。如果callback返回值为null，则删除该元素。	var temp=$.map(arrB,function(n){
		return n>7?n+10:n-10;
		});//给大于7的元素加上10，否则减去10。
		//temp为Array，值为-6,-5,-4,-3,18,19,20。
		arrB不发生变化。

注：假设var arrA=[1,2,3],arrB=[4,5,6,7,8,9,10];//arrA.length等于3，arrB.length等于7。各用例之间互相独立。		

4、字符串对象(String)

在前面各个章节中，已经多次使用字符串对象，如：var thisBookName="New Way to Programming";中的thisBookName实际上就是一个字符串对象，操作也仅仅限于字符串拼接等简单操作，没有提出字符串对象概念，以及更多与其相关的操作。和其他语言相比，学习Web编程要更加在意字符串操作，HTML内容、属性等操作都表现为字符串。

⑴ String对象申明

下面列出了新的字符串申明方式，实际工作中，一般采用方式3或方式5申明一个新的字符串。

String对象申明方式

方式1：var strName=String("abcd");

方式2：var strName=new String("abcd");

方式3：var strName="abcd"

方式4：var strName=new String();

方式5：var strName="";

备注：strName是数组名称，与变量命名规则相同。方式4、方式5为空字符串。abcd代表字符组合。

⑵ String对象属性和方法

String最常用的属性是length，其含义是字符串长度。和Array对象不同，改变length大小，不能调整字符串长度，可以认为length是只读属性，即只可以获取字符串长度，而不能改变字符串长度。

例程4-55

```
第1行  <SCRIPT LANGUAGE="JavaScript">
第2行      var strTemp=new String("abcd");
第3行      alert(strTemp.length);//显示为4
第4行  </SCRIPT>
```

String对象的操作方法较多，大致可以分为：字符串拼接(String.concat)、查找匹配(String.indexOf、String.lastIndexOf、String.search、String.match等)、片段截取(String.charAt、String.charCodeAt、String.substr、String.substring、String.slice)、替换(String.replace)、转换(String.lowerCase、String.upperCase、String.localeLowerCase、String.localeUpperCase等)以及一系列HTML标记生成等。

表8：String操作方法

方法	描述	用例
String.concat(str1,str2,……,strN)	连接字符串，相当于字符串+运算符，将str1等追加到String对象内容尾部。	var str1="ABC",str2="XYZ"; str1.concat(str2);//str1变为"ABCXYZ"
String.indexOf(str1)	检索字符串，检索str1在String对象中的位置。	var testStr="I love China.",str1="love"; var strPos=testStr.indexOf(str1);//strPos值为2
lastIndexOf(str1)	从后向前搜索字符串。	var testStr="人民警察为人民",str1="人民"; var strPos=testStr.indexOf(str1);//strPos值为5
search()	检索与正则表达式相匹配的值，参考“正则表达式”章节。	
match()	找到一个或多个正则表达式的匹配，参考“正则表达式”章节。	

localeCompare()	用本地特定的顺序来比较两个字符串。	

charAt(intPos)	返回在指定位置(intPos)的一个字符。	var testStr="I love China.";
		var singleChar=testStr.charAt(4);
		//singleChar为v。

charCodeAt(intPos)	返回在指定位置(intPos)字符的ASCII/Unicode编码。	var testStr="I love China.";
		var charNo=testStr.charCodeAt(4);
		//charNo为118，即v的ASCII值。

substr(intPos,intCount)	从指定位置(intPos)开始提取指定数目(intCount)的字符，如省略intCount则从指定位置(intPos)开始，截取到末尾。	var testStr="I love China.";
		var sonStr1=testStr.substr(2,4);
		//sonStr1为love
		var sonStr2=testStr.substr(2);
		//sonStr2为love China.

substring(intPos1,intPos2)	提取字符串中两个指定位置(intPos1,intPos2)之间的字符，如省略intPos2则从intPos1开始全部字符。	var testStr="I love China.";
		var sonStr1=testStr.substring(2,6);
		//sonStr1为love
		var sonStr2=testStr.substring(2);
		//sonStr2为love China.

slice(intPos1,intPos2)	与substring()相似。提取intPos1到intPos2之间的字符串片断，支持负数，如-1表示最后一个字符。如省略intPos2则返回从intPos1开始的全部字符。	var testStr="I love China.";
		var sonStr1=testStr.slice(2,6);
		//sonStr1为love
		var sonStr2=testStr.slice(2);
		//sonStr2为love China.
		var sonStr3=testStr.slice(-6,-1);
		//sonStr3为China

fromCharCode(intVal)	从字符编码创建一个字符串。	var singleChar=String.fromCharCode(118); //singleChar为v。
replace(str1,str2)	查找str1，并替换为str2，str1可为正则表达式。	var testStr="美丽的祖国，美丽的人民！"; var newStr=testStr.replace("美丽","可爱"); //newStr为"可爱的祖国，美丽的人民！"

split(str1)	用str1作为分隔符，将字符串分割为字符串数组。str1可以为正则表达式。如省略str1则数组为每个字符。	var testStr="美丽的祖国，美丽的人民！";
		var newStr1=testStr.split("的");
		//newStr1为美丽,祖国,美丽,人民 var newStr2=testStr.split("的");
		//newStr2为美,丽,的,祖,国,美,丽,的,人,民

toLocaleLowerCase()	把字符串转换为小写。	
toLocaleUpperCase()	把字符串转换为大写。	
toLowerCase()	把字符串转换为小写。	
toUpperCase()	把字符串转换为大写。	
toSource()	代表对象的源代码。	
toString()	返回字符串。	
valueOf()	返回某个字符串对象的原始值。	

⑶ String对象例程

String对象的属性和方法应用很多。

例程4-56

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        function engDig2chn(engNum){
第3行            return "零一二三四五六七八九".charAt(engNum);
```

```
第4行                }
第5行          </SCRIPT>
```

在例程4-56中， "零一二三四五六七八九"构成一个String对象，其charAt()方法用于取得指定位置的字符，位置指定由函数engDig2chn的参数engNum提供。如engNum为0，则chatAt(0)返回“零”，以此类推。

例程4-57

```
第1行    function changeImg(){
第2行          var imgFullFilename=$("#myImg").attr("src");//取出包含位置的图片文件字符串
第3行
第4行          var lastSlashPos=imgFullFilename.lastIndexOf("/");//最后一个斜杠位置
第5行
第6行          var imgFilename=imgFullFilename.substring(lastSlashPos+1);//取出图片文件名，不包含位置;
第7行          var imgPath=imgFullFilename.substring(0,lastSlashPos+1);//图片位置
第8行
第9行          //取得图片编号
第10行         var periodPos=imgFilename.indexOf(".");//扩展名与主文件名之间句点的位置
第11行         var currentImgNo=imgFilename.substring(0,periodPos);//正在显示的图片编号
第12行
第13行         var nextImgNo=parseInt(currentImgNo)+1;//计算出下一个图片编号
第14行
第15行         //nextImgNo大于10则让nextImgNo等于1，相当于从头开始
第16行         if(nextImgNo>10){
第17行                nextImgNo=1;
第18行         }
第19行
第20行         var imgSRC=imgPath+nextImgNo.toString()+".gif";//合成IMG标志的src属性
第21行
第22行         $("#myImg").attr("src",imgSRC);//改变click图片的src属性
第23行         $("#myImg").attr("ImgNo",nextImgNo);//修改ImgNo属性，保存当前编号
第24行   }
```

例程4-57是对前面相关例程的调整，其目的是通过显示在IMG标记中的src属性，计算出相关数据，如：图片位置，图片编号等。在例程中，首先取得IMG标记的src属性，并保存在imgFullFilename中。在此基础上，计算出最后一个斜杠位置，其前为位置信息(imgPath)，其后为图片文件名(imgFilename)。在图片文件名的基础上，通过计算句点位置(periodPos)，其前为图片编号(currentImgNo)，其后为扩展文件名。

例程4-57实现了图片切换功能，相对于changeImg()函数的其他实现方式，这种方式可能稍显复杂，其主要目的是彰显String对象各种方法的应用。另外，图片编号已在src属性中体现，完全可以不通过其他途径即可实现计算出下一个图片编号。

例程4-58

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行          alert(dec2Hex(122));//样例，122转换为十六进制，结果为7A
第3行
第4行          function dec2Hex(Num){
第5行                var rtnResult="";
第6行                while(Num>0){
第7行                      var modNum=Num%16;//除以16的余数
第8行
```

```
第9行                    if(modNum<10)
第10行                       rtnResult=modNum.toString()+rtnResult;//小于10
第11行                   else//大于10，其中"A".charCodeAt(0)表示A的ASCII值
第12行                       rtnResult=String.fromCharCode("A".charCodeAt(0)+modNum-10)+rtnResult;
第13行
第14行                   Num=(Num-Num%16)/16;//Num值变化，即得到除以16的商
第15行               }
第16行               return rtnResult;
第17行           }
第18行   </SCRIPT>
```

当余数小于10时，正常累加即可。当大于10时，分别对应到A，B，……，F，其值对应ASCII码分别是"A".charCodeAt(0)加上相应的值，如：B对应余数11，相对于A大1，即余数减去10即可，即"A".charCodeAt(0)+modNum-10，然后通过String.fromCharCode()即可转换得到相应的字母。

String对象方法中，replace、search、match支持正则表达式。正则表达式用来描述或者匹配一系列符合约定句法规则的字符串。正则表达式具有灵活性、逻辑性和功能性都非常强的特点，可以迅速地用极简单的方式达到字符串的复杂控制。相关内容请参见“正则表达式”章。

第八节　更多了解

1、单双引号

2、变量命名