即用即查——JavaScript 核心对象参考手册

丛书特点：直击核心技术 解惑应用开发

- 案头必备、内容实用
- 排版精准、图文并茂
- 光盘中提供了索引，方便快速查找

书名：即用即查——JavaScript 核心对象参考手册
作者：韩延峰　编著
书号：978-7-115-16140-6
定价：49
开本：185×260　1/16
印张：27.75
总页码：442
印刷色别：单
有无光盘：1CD
选题类别：网络开发
预计出版时间：2007 年 5 月

本书讲述：

- 15 个 JavaScript 对象
- 372 个属性方法
- 307 个实例源文件
- 7 个附录

内容涵盖：

- JavaScript 开发基础：51 个实例
- 创建和使用自定义对象：7 个实例
- Date 对象：23 个实例
- String 对象：27 个实例
- Math 对象：26 个实例
- 数组与 Array 对象：18 个实例
- RegExp、Object、Error、Number 对象：22 个实例
- Form 对象：14 个实例
- Window 对象：39 个实例
- 事件及事件响应：19 个实例
- ActiveX 技术：5 个实例
- 文件访问对象：31 个实例
- 数据库访问：25 个实例

技术深度：（初、中）

目标受众：

□ 从事 Web 应用开发的人员；
□ JavaScript 初学者；
□ 网页设计与制作人员；
□ 网页制作爱好者；
□ 大中专院校爱好 IT 的学生。

内容提要：

本书是一本 JavaScript 核心对象的速查手册，其中穿插丰富的实例。本书内容涵盖了 JavaScript 基础知识、常用对象、DOM 对象以及 ActiveX 等高级应用的各方面知识。涉及 Date 对象、String 对象、Math 对象、Array 对象、Function 对象、Number 对象、Object 对象、Form 表单、Document 对象、Window 对象等。

本书适合 JavaScript 初学者、JavaScript 程序员以及其他 Web 应用程序工程技术人员阅读，同时也特别适合想学习 Ajax 技术的读者。

目录：

样章：

目录：

上篇　JavaScript 开发入门

下篇　ActiveX 插件技术

样章：

第 12 章

数组与 Array 对象：创建和操作数组

如第三章所介绍，在程序中数据是存储在变量中的，但是，如果数据量很大，比如几百个学生的成绩，此时再逐个定义变量来存储这些数据就显得异常繁琐，如果通过数组来存储这些数据就会使这一过程大大简化。在编程语言中，数组是专门用于存储有序数列的工具，也是最基本、最常用的数据结构之一。在 JavaScript 中，Array 对象专门负责数组的定义和管理，本章将详细地介绍数组的作用和 Array 对象的各个属性和方法。

12.1 数组概述

可以把数组看作一行表格，该表格的每一个单元格中都可以存储一个数据，而且各个单元格中存储的数据可以不同。这些单元格被称为数组元素，每一个数组元素都有一个索引号，通过索引号可以方便地引用数组元素。

12.1.1 数组索引

数组索引是数组元素的标记，在 JavaScript 中，数组索引从 0 开始，最大索引为数组元素的个数减 1。数组索引与数组元素的对应关系如图 12.1 所示。

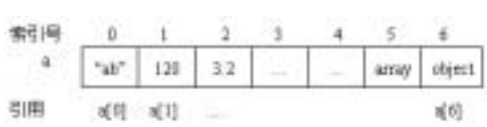

图 12.1　数组元素与数组索引

图中 a 为数组名称，该数组的长度为 7，最大索引为 6。因此如果定义了一个长度为 n 的数组，那么该数组的最大索引为 n-1。在 JavaScript 中，如果使用数组时指定的索引超过了该数组的最大索引，系统并不会提示错误，而是自动增加数组的长度，以适应存储要求。

通过索引引用数组元素的一般格式为：

```
array[index]
```

参数说明如下。

□ array：必选项，数组名称。

□ index：必选项，数组索引。

在引用数组元素时，一般配合使用数组索引和 for 语句以提高效率，下面的代码演示了如何通过数组索引来访问数组。

```
<script>
    //定义数组
    var arr=new Array(3);
    //给数组元素赋值
    arr[0]="first";
    arr[1]="second";
    //利用for语句赋值，超出索引范围
    for(var i=2;i<5;i++)
    {
        arr[i]=i;
```

```
    }
    //利用for语句输出所有数组元素
    for(var i in arr)
    {
       document.write(arr[i]);
       document.write("<br>");
    }
</script>
```

这段代码中，首先定义了一个长度为 3 的数组，然后给数组的前两个元素逐个赋了值，之后又使用 for 语句给其他元素赋了值，并且指定的索引超出了定义的范围，最后通过 for-in 语句输出了数组中的全部数据。运行这段代码，可以看到图 12.2 所示的页面效果。

图 12.2　数组索引

在 JavaScript 中可以通过多种方法来创建数组，下面介绍常用的几种。

12.1.2　通过 Array 对象创建数组

这是最常用的一种创建方法，这种创建方法有以下两种格式。

```
var arrname=new Array([length])
var arrname=new Array([arg1[,arg2[,…[,argn]]]])
```

参数说明如下。

□ arrname：必选项，所定义的数组的名称。

□ length：可选项，整数值，用于指定所定义的数组的长度。

□ arg1、arg2、…、argn：可选项，任意类型的数据，初始化数组的数据。

如果定义数组时没有指定任何参数，则创建一个空数组，该数组的内容可以在以后的操作中根据需要添加；如果定义数组时，只指定了一个参数且该参数为正整数，则会返回一个以该整数长度的数组；如果同时指定了多个参数，则会返回一个存储有指定参数的数组，该数组的长度由参数的个数确定。

下面的代码演示了利用 Array 对象创建数组时的这三种不同的情况。

```
<script>
    var arr1=new Array();
    var arr2=new Array(4);
    var arr3=new Array(1,2,3,"a","b");
    arr1[0]="add 1";
    arr1[1]="add 2";
    for(var i=0;i<4;i++)
    {
       arr2[i]=i;
    }
    writearr("arr1",arr1);
    writearr("arr2",arr2);
    writearr("arr3",arr3);
    //自定义函数，输出数组中所有元素
    function writearr(strinfo,arr)
    {
        document.write(strinfo+":");
        for(var i=0;i<arr.length;i++)
        {
```

```
                document.write(arr[i]);
                document.write("    ");
            }
            document.write("<br>");
        }
    </script>
```

运行这段代码可以看到图 12.3 所示的页面效果。

图 12.3　代码执行效果

12.1.3　自定义数组构造函数创建数组

这种方法需要先定义一个构造函数，在构造函数内实现数组的创建和初始化，这时要借助于 this 指针，具体的实现过程可以参考下面的实例。

```
    <script>
        //自定义构造函数
        function myArray(n,initvalue)
        {
            for(var i=0;i<n;i++)
                this[i]=initvalue;
            this.length=n;
        }
        var strhead="<input type='text' size='4' name='score";
        var strend="'>";
        var num=20
        //利用构造函数创建数组
        var arr=new myArray(num,strhead);
        document.write("<form>");
        //修改数组元素并输出
        for(var i=0;i<num;i++)
        {
            if (i%5==0)
                document.write("<br>");
            arr[i]+=i+strend;
            document.write(arr[i]);
        }
        document.write("<br><input type='button' value='提交'>");
        document.write("</form>");
    </script>
```

这段代码中自定义了一个构造函数，该构造函数含有两个参数，第一参数用于指定数组的长度，第二个参数用于指定数组元素的初始值。运行这段代码可以看到图 12.4 所示的页面效果。

图 12.4　自定义数组构造函数演示

可以看到：通过自定义构造函数来创建数组具有更大的灵活性，可以方便简洁地实现一些特殊功能。

12.1.4　通过其他对象的方法获取创建数组

一些对象的某些方法的返回值就是一个数组，比如 String 对象的 split 方法，该方法就可以将指定的字符串划分为若干部分，存储到数组中并返回。通过这种方式得到的数组可以和前面两种方法创建的数组一样使用。下面的代码演示了利用 String 对象的 split 方法创建数组的过程。

```
<script>
    var str="this is a test";
    //获取数组
    var arr=str.split(" ");
    //操作数组
    for(var i=0;i<arr.length;i++)
    {
        arr[i]+="_";
    }
    arr[arr.length]="!"
    //输出数组
    for(var i in arr)
       document.write(arr[i]);
</script>
```

这段代码通过调用 String 对象的 split 方法获取了一个数组 arr，然后利用 for 循环对数组中所有元素的值进行了修改，之后又在数组的末尾新添加了一个元素并赋值为“ !”，最后输出了数组中的全部数据。运行这段代码，执行效果如图 12.5 所示。

图 12.5　通过 split 方法创建数组

12.2　Array 对象的属性

Array 对象的属性有 3 个：length 属性、constructor 属性和 prototype 属性。本节将简单介绍一下 length 属性和 prototype 属性的应用。对于 constructor 属性，可以参考前面介绍的 Date 对象的 constructor 属性。

12.2.1　length 属性：返回数组的长度

【功能说明】所谓数组的长度是指数组中数组元素的个数，而且其数值会随着数组元素的增减而自动改变，因此，利用该属性可以方便地实现对数组的遍历。

【基本语法】array.length

其中，array 为数组名称。

【实例演示】

```
<script>
    var arr=new Array(1,2,3,4,5,6,7,8);
    with (document)
    {
        write("数组长度:"+arr.length);
        write("<br>新增一个元素")
        arr[arr.length]=arr.length+1;
        write("<br>数组长度:"+arr.length);
        write("<br>奇数位元素：");
        for(var i=0;i<arr.length;i+=2)
        {
            write(arr[i]+",");
        }
        write("<br>偶数位元素：");
        for(var i=1;i<arr.length;i+=2)
        {
            write(arr[i]+",");
        }
    }
</script>
```

运行这段代码可以看到图 12.6 所示的页面效果。

从本例中可以看到，数组的 length 会随数组中元素个数的增减而自动变化，利用 length 属性和 for 语句可以方便地遍历数组中的所有元素。

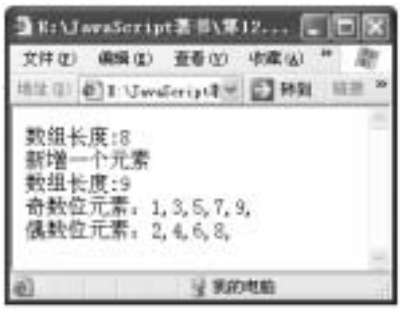

图 12.6　length 属性演示

12.2.2　prototype 属性：将新定义的属性或方法添加到 Array 对象中

【功能说明】该属性是所有 JavaScript 对象所共有的属性，和 Date 对象的 prototype 属性一样，其作用为将新定义的属性或方法添加到 Array 对象中，然后，该对象的实例就可以调用该属性或方法。

【基本语法】Array.prototype.methodName=functionName

参数说明如下。

□ methodName：必选项，新增方法的名称。

□ functionName：必选项，要添加到对象中的函数名称。

【实例演示】

```
<script>
  //添加一个属性，用于统计删除的元素个数
  Array.prototype.removed=0;
  //添加一个方法，用于删除指定索引的元素
  Array.prototype.removeAt=function(index)
  {
    if(isNaN(index)||index<0)
        {return false;}
    if(index>=this.length)
        {index=this.length-1}
    for(var i=index;i<this.length;i++)
    {
      this[i]=this[i+1];
```

```
        }
        this.length-=1
        this.removed++;
    }
    //添加一个方法，输出数组中的全部数据
    Array.prototype.outPut=function(sp)
    {
        for(var i=0;i<this.length;i++)
        {
            document.write(this[i]);
            document.write(sp);
        }
        document.write("<br>");
    }
    //定义数组
     var arr=new Array(1,2,3,4,5,6,7,8,9);
     //测试添加的方法和属性
     arr.outPut(" ");
     document.write("删除一个数据<br>");
     arr.removeAt(2);
     arr.outPut(" ");
     arr.removeAt(4);
     document.write("删除一个数据<br>");
     arr.outPut(" ")
     document.write("一共删除了"+arr.removed+"个数据");
</script>
```

这段代码利用 prototype 属性分别向 Array 对象中添加了两个方法和一个属性，分别实现了删除指定索引处的元素、输出数组中的所有元素和统计删除元素个数的功能。运行这段代码可以看到图 12.7 所示的页面效果。

图 12.7　prototype 属性演示

12.3　Array 对象的方法

Array 对象常用的方法有 13 个，依次介绍如下。

12.3.1　concat 属性：连接其他数组到当前数组末尾

【功能说明】该方法的作用是把当前数组和指定的数组相连接，然后返回一个新的数组，该数组中含有前面两个数组的全部元素，其长度为两个数组的长度之和。

【基本语法】array1.concat(array2)

参数说明如下。

□ array1：必选项，数组名称。

□ array2：必选项，数组名称，该数组中的元素将被添加到数组 array1 中。

【实例演示】

```
<script>
    var array1=new Array(1,2,3,4,5,6,7);
```

```
    var array2=new Array(8,9,10);
    var array=array1.concat(array2);
    //自定义函数，输出数组中所有数据
    function writeArr(arrname,sp)
    {
      for(var i=0;i<arrname.length;i++)
      {
          document.write(arrname[i]);
          document.write(sp);
      }
      document.write("<br>");
    }
    document.write("数组1：");
    writeArr(array1,",");
    document.write("数组2：");
    writeArr(array2,",");
    document.write("数组3：");
    writeArr(array,",");
</script>
```

这段代码定义了两个数组 array1 和 array2，然后把这两个数组连接并将值赋给数组 array。运行这段代码可以看到图 12.8 所示的页面效果。

图 12.8　concat 方法演示

12.3.2　jion 方法：将数组元素连接为字符串

【功能说明】该方法与 String 对象的 split 方法的作用相反，该方法的作用是将数组中所有元素连接为一个字符串，如果数组中的元素不是字符串，则该元素将首先被转化为字符串，各个元素之间可以以指定的分隔符进行连接。

【基本语法】array.jion(separator)

参数说明如下。

□ array：必选项，数组的名称。

□ separator：必选项，连接各个元素之间的分隔符。

【实例演示】

下面的代码对比了 split 方法和 jion 方法。

```
<script>
    var str1="this ia a test";
    var arr=str1.split(" ");
    var str2=arr.join(",");
    with(document){
      write(str1);
      write("<br>分割为数组，数组长度"+arr.length+",重新连接如下：<br>");
      write(str2);
    }
</script>
```

这段代码首先使用 split 方法以“　”（空格）为分隔符将字符串分割存储到数组中，然后调用 join 方法以“，”（逗号）为分隔符，将数组中的各个元素重新连接为一个新字符串。运行这段代码可以看到图 12.9 所示的页面效果。

图 12.9　join 方法演示

12.3.3　pop 方法：删除数组中最后一个元素

【功能说明】该方法将数组中的最后一个元素删除并返回该元素，调用该方法后，数组的长度将减小 1。如果数组为空，则该方法的返回值为 undefined。

【基本语法】array.pop()

【实例演示】

```
<script>
    var arr=new Array(1,2,3,4,5,6,7,8,9);
    with (document)
    {
        write(arr.join(","));
        write("<br>删除元素"+arr.pop()+"<br>");
        write("删除元素"+arr.pop()+"<br>");
        write(arr.join(","));
    }
</script>
```

运行这段代码，执行结果如图 12.10 所示。

图 12.10　pop 方法演示

12.3.4　push 方法：将指定的数据添加到数组中

【功能说明】该方法可以将所指定的一个或多个数据添加到数组中，该方法的返回值为添加新数据后数组的长度。

【基本语法】array.push([data1[,data2[,…[,datan]]]])

参数说明如下。

□ array：必选项，数组名称。

□ data1、data2、datan：可选参数，将被添加到数组中的数据。

如果 data1 到 datan 中的某一参数为数组，则该数组中的所有元素将被添加到数组中。

【实例演示】

下面的代码演示了如何利用 push 方法向数组中添加新数据。

```
<script>
    var arr=new Array();
    document.write("向数组中写入数据：");
    //单个数据写入数组
    for (var i=1;i<=4;i++)
```

```
    {
        var data=arr.push(Math.ceil(Math.random()*10));
        document.write(data);
        document.write("个,");
    }
    document.write("<br>");
    //批量写入数组
    var data=arr.push("a",3.14,"hello");
    document.write("批量写入，数组长度已为"+data+"<br>");
    var newarr=new Array(1,2,3,4,5);
    document.write("向数组中写入另一个数组<br>");
    //写入新数组
    arr.push(newarr);
    document.write("全部数据如下:<br>");
    document.write(arr.join(","));
</script>
```

这段代码分别使用 push 方法向数组中逐个和批量添加了数据，运行这段代码，执行结果如图 12.11 所示。

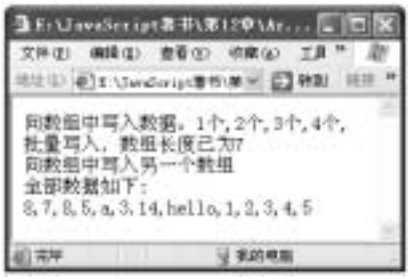

图 12.11　push 方法演示

12.3.5　reverse 方法：反序排列数组中的元素

【功能说明】该方法可以将数组中的元素反序排列，数组中所包含的内容和数组的长度不会改变。

【基本语法】array.reverse()

其中，array 为数组的名称。

【实例演示】

```
<script>
    var arr=new Array(1,2,3,4,5,6,7,8,9);
    with (document)
    {
        write("数组为:");
        write(arr.join(","));
        arr.reverse();
        write("<br>反序后:")
        write(arr.join(","));
    }
</script>
```

运行这段代码，执行结果如图 12.12 所示。

图 12.12　reverse 方法演示

12.3.6 shift 方法：删除数组中的第一个元素

【功能说明】该方法同 pop 方法类似，不同的是该方法删除的是数组中的第一个元素，其返回值为该元素的值。

【基本语法】array.shift()

其中，array 为数组的名称，下面的代码演示了 shift 方法的用法和作用。

【实例演示】

```
<script>
    var arr=new Array(1,2,3,4,5,6,7,8,9);
    with (document)
    {
        write(arr.join(","));
        write("<br>删除元素"+arr.shift()+"<br>");
        write("删除元素"+arr.shift()+"<br>");
        write(arr.join(","));
    }
</script>
```

运行这段代码，执行结果如图 12.13 所示。

图 12.13　shift 方法演示

12.3.7 slice 方法：获取数组中的一部分数据

【功能说明】该方法从数组中提取一部分数据，并将这部分数据作为一个数组返回。

【基本语法】array.slice(start[,end])

参数说明如下。

□ array：必选项，数组名称。

□ start：必选项，获取数据的起始索引位置，从 0 开始。

□ end：可选项，获取数据的结束位置，从 0 开始。

该方法返回的数据中不包括 end 索引所对应的数据；如果 start（end）的值为负值，则其值将被自动替换为 start（end）+length（length 为数组长度）；如果没有指定 end 值，则返回从 start 开始到数组末尾的所有元素。

【实例演示】

```
<script>
    var arr=new Array(1,2,3,4,5,6,7,8,9,10);
    writeArr("原始数组arr",arr);
    writeArr("arr.slice(2,5)提取片断",arr.slice(2,5));
    writeArr("arr.slice(-8,-1)提取片断",arr.slice(-8,-1));
    writeArr("arr.slice(1)提取片断",arr.slice(1));
    //自定义函数输出提示信息和数组元素
    function writeArr(str,array)
    {
        document.write(str+":");
        document.write(array.join(","));
        document.write("<br>");
    }
</script>
```

运行这段代码可以看到图 12.14 所示的页面。

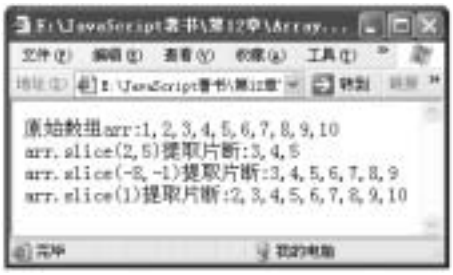

图 12.14　slice 方法演示

12.3.8　sort 方法：对数组中的元素进行排序

【功能说明】该方法对数组中的所有元素按 Unicode 编码进行排序，并返回经过排序后的数组。sort 方法默认按升序进行排列，但也可以通过指定对比函数来实现特殊的排序要求，对比函数的格式如下：

```
comparefunction(arg1,arg2)
```

其中，comparefunction 为排序函数的名称，该函数必须包含两个参数 arg1 和 arg2，分别代表了两个将要进行对比的字符。该函数的返回值决定了如何对 arg1 和 arg2 进行排序。

□ 返回值为负，则 arg2 将排在 arg1 的后面。

□ 返回值为 0，arg1、arg2 视为相等。

□ 返回值为正，则 arg2 将排在 arg1 的前面。

【基本语法】array.sort([cmpfun(arg1,arg2)])

参数说明如下。

□ array：必选项，数组名称。

□ cmpfun：可选项，比较函数。

□ arg1，arg2：可选项，比较函数的两个参数。

【实例演示】

下面的代码演示了如何使用 sort 方法对数组中的数据进行排序。

```
<script>
    var arr=new Array(2,5,3,20,1,"b","x","B","X");
    writeArr("排序前",arr);
    writeArr("升序排列",arr.sort());
    writeArr("降序排列,字母不分大小写",arr.sort(desc));
    writeArr("严格降序排列",arr.sort(desc1));
    //自定义函数输出提示信息和数组元素
    function writeArr(str,array)
    {
        document.write(str+":");
        document.write(array.join(","));
        document.write("<br>");
    }
    //按降序排列,字母不区分大小写
    function desc(a,b)
    {
        var a=new String(a);
        var b=new String(b);
        //如果a大于b，则返回－1，所以a排在前b排在后
        return -1*a.localeCompare(b) ;
    }
    //严格降序
    function desc1(a,b)
    {
            var stra=new String(a);
            var strb=new String(b);
            var ai=stra.charCodeAt(0);
            var bi=strb.charCodeAt(0);
            if( ai>bi )
               return -1;
            else
```

```
            return 1;
        }
    </script>
```

这段代码中定义了两个对比函数，其中 desc 进行降序排列，但字母不区分大小写；desc1 进行严格降序排列。运行这段代码，执行结果如图 12.15 所示。

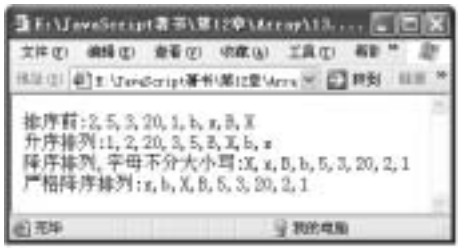

图 12.15　sort 方法演示

12.3.9　splice 方法：删除或替换数组中部分数据

【功能说明】该方法可以通过指定起始索引和数据个数的方式，删除或替换数组中的部分数据。该方法的返回值为被删除或替换掉的数据。

【基本语法】array.splice(start,count[,data1[,data2,[,…[,datacount]]]])

参数说明如下。

□ array：必选项，数组名称。

□ start：必选项，整数，起始索引。

□ count：必选项，整数，要删除或替换的数组的个数。

□ data：可选项，用于替换指定数据的新数据。

如果没有指定 data 参数，则该指定的数据将被删除；如果指定了 data 参数，则数组中的数据将被替换。

【实例演示】

```
<script>
    var arr=new Array(0,1,2,3,4,5,6,7,8,9,10);
    var rewith=new Array("a","b","c");
    var tmp1=arr.splice(2,4,rewith);
    with(document)
    {
        writeArr("替换了4个数据",tmp1);
        writeArr("替换为：",rewith);
        writeArr("替换后",arr);
        var tmp2=arr.splice(5,2);
        writeArr("删除2个数据",tmp2);
        writeArr("替换后",arr);      }
   //自定义函数输出提示信息和数组元素
   function writeArr(str,array)
   {
       document.write(str+":");
       document.write(array.join(","));
       document.write("<br>");
   }
</script>
```

这段代码分别演示了如何使用 splice 方法替换和删除数组中指定数目的数据。运行这段代码，执行结果如图 12.16 所示。

图 12.16　splice 方法演示

12.3.10　unshift 方法：在数组前面插入数据

【功能说明】该方法与 shift 方法的作用相反，该方法在数组的开始插入一个或多个数据，返回值为增加了新插入数据之后的数组长度。

【基本语法】array.unshift([data1[,data2[,…[,datan]]]])

参数说明如下。

□ array：必选项，数组名称。

□ data1 到 datan：可选项，要插入到数组开始的数据。

【实例演示】

```
<script>
    var arr=new Array();
    //单个数据写入数组
    for (var i=1;i<=4;i++)
    {
        arr.unshift(i);
    }
    document.write("<br>");
    //批量写入数组
    arr.unshift("a",3.14,"hello");
    var newarr=new Array(1,2,3,4,5);
    //写入新数组
    arr.unshift(newarr);
    document.write("全部数据如下:<br>");
    document.write(arr.join(","));
</script>
```

运行这段代码，执行结果如图 12.17 所示。

图 12.17　unshift 方法演示

12.3.11　toString 方法：返回一个包含数组中全部数据的字符串

【功能说明】toString 方法是所有 JavaScript 对象所共有的方法，对于 Array 对象来说，该方法与 join 方法的功能类似，它将数组中的所有元素连接为一个字符串，各个元素之间

使用逗号（“,”）连接。

【基本语法】array.toString()

【实例演示】

```
<script>
    var arr=new Array(1,2,3,4,"this ia ","a test");
    document.write(arr.toString());
</script>
```

运行这段代码，执行结果如图 12.18 所示。

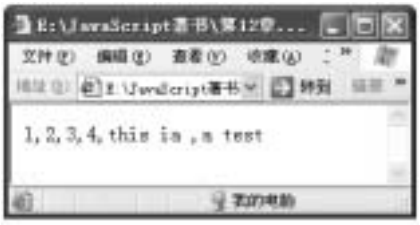

图 12.18　toString 方法演示

12.4　进一步讨论：二维数组的实现

许多编程语言中都提供定义和使用二维或多维数组的功能。JavaScript 通过 Array 对象创建的数组都是一维的，但是可以通过在数组元素中使用数组来实现二维数组。如果把一维数组看作为一行数据表格，那么，二维数组就可以看作为多行数据表格，因此，二维数组常用来存储含有多个数据项的条目性数据内容，比如表 12.1 所示的数据表中的数据就适合使用二维数组进行存储。

表 12.1　二维数据表

01042403	韩永强	环境保护导论	88
01042404	黄大伟	数值分析	85
01042405	黎明	人工举升	90
01042406	李高飞	高等渗流力学	79
01042407	王小虎	高等流体力学	96
01042408	田琳	数值传热学	85

下面的代码创建了一个二维数组的构造函数。

【实例演示】

```
function Array2(m,n)
{
    for(var i=0;i<m;i++)
    {
        var arrtmp=new Array(n);
        this[i]=arrtmp;
    }
    this.length=m;
}
```

这个构造函数可以根据指定的参数创建一个 m 行 n 列的二维数组。下面通过一个具体的例子来看一下如何使用和操作二维数组存储数据。

```
<script>
    var counter=0;
    //二维数组构造函数
    function Array2(m,n)
    {
        var arr=new Array(m);
```

```
        for(var i=0;i<m;i++)
        {
           var arrtmp=new Array(n);
           arr[i]=arrtmp;
        }
        return arr;
    }
    var arrstudent=new Array2(1,4)
    var arr=new Array(4);
    /*  将输入的数据存入数组
       同时按成绩降序排列显示
    */
    function toarr()
    {
          if(event.keyCode==13)
          {
               var id=document.forms[0].stunum.value;
               var name=document.forms[0].stuname.value;
               var sclass=document.forms[0].stuclass.value;
               var score=document.forms[0].stumark.value;
                arrstudent[counter]=new Array(id,name,sclass,score);
                counter++;
                document.forms[0].stunum.value="";
                document.forms[0].stuname.value="";
                document.forms[0].stumark.value="";
                document.forms[0].stunum.focus();
                listAll();
          }
    }
    //以表格的形式输出数组中的数据
    function listAll()
    {
        sortBy(3);//按成绩排序
        var str="<table cellpadding='0' cellspacing='0' border='1'>";
        for(var i=0;i<counter;i++)
        {
            str+="<tr>"
            for(var j=0;j<4;j++)
            {
                str+="<td>"+arrstudent[i][j]+"</td>";
            }
            str+="</tr>";
        }
        str+="</table>";
        document.getElementById("list").innerHTML=str;
    }
    /*
        二维数组的排序操作
        可以按数组中的任意一列进行排序，降序
    */
    function sortBy(index)
    {
         for(var i=0;i<arrstudent.length-1;i++)
         {
              for(var j=i+1;j<arrstudent.length;j++)
              {
                        if(arrstudent[i][index].localeCompare(arrstudent[j][index])==-1)
                        {
                            var temp=new Array(4);
                           //交换两行
                            temp=arrstudent[i];
                            arrstudent[i]=arrstudent[j];
                            arrstudent[j]=temp;
                        }
              }
         }
    }
</script>
<form>
    <table cellpadding="0" cellspacing="0">
       <tr bgcolor="#CCCCCC" bordercolor="#333333">
            <th>学号</th><th>姓名</th><th>课程</th><th>成绩</th>
     </tr>
     <tr>
            <td><input type="text" name="stunum" size="10" /></td>
            <td><input type="text" name="stuname" size="10" /></td>
            <td> <select name="stuclass">
                    <option value="数值分析">数值分析</option>
                    <option value="人工举升">人工举升</option>
                    <option value="数值传热学">数值传热学</option>
                    <option value="高等流体力学">高等流体力学</option>
```

```
                    </select>
                </td>
                <td><input type="text" name="stumark" size="4"    onKeyDown="toarr()"/></td>
            </tr>
            <tr>
                <td align="center" colspan="4">
                    <div id="list" onClick="listAll()" style="background:#CCCCCC">列出所有成绩</div>
                </td>
            </tr>
        </table>
    </form>
```

这段代码中定义了 4 个函数：Array2、toarr、listAll 和 sortBy，分别实现了构造二维数组、将收入数据存入二维数组、以表格形式显示存储数据和对二维数组中的数据进行排序的功能。运行这段代码，执行效果如图 12.19 所示。

分别输入学号、姓名并选择课程，输入成绩后按下回车键，该条信息（包括学号、姓名、课程和成绩 4 项）将被存入数组中并显示在下面列表中，之后学号的文本框将自动获取焦点，如图 12.20 所示。

图 12.19　程序运行效果

图 12.20　成绩输入后

继续按学号顺序输入多条数据，可以看到输出列表中的数据与输入顺序无关，而是以成绩进行降序排序，如图 12.21 所示。

图 12.21　数据输出列表

第 16 章

Window 对象：访问和控制浏览器窗口

Window 对象是 DOM 对象模型的最顶层对象，代表了浏览器中用于显示文档内容的窗口，通过该对象可以访问 DOM 对象模型中的所有对象。Window 对象和 Javascript 的内置对象一样，使用的时候无需手动创建，只要在 HTML 文档或者 Web 文档中使用了<Body>标签或者<frame>标签，系统就会自动创建一个 Window 对象。

16.1 Window 对象的方法

16.1.1 alert 方法：弹出一个警告对话框

【功能说明】使用 Alert 方法可以弹出一个含有制定内容的警告对话框，该对话框中还包含有一个叹号图标和一个“ 确定 ”按钮。

【基本语法】window.alert(message)

其中，message 为任意有效的字符串表达式，该字符串指定了要在对话框上显示的内容。

【实例演示】下面的代码利用 Alert 方法弹出了一个含有提示信息的对话框。

```
<body>
    <script>
        var strmsg="Window对象的Alert方法测试！"；
        window.alert(strmsg)；
    </script>
</body>
```

运行这段代码可以看到图 16.1 所示的对话框。

图 16.1　Alert 方法演示

16.1.2 confirm：弹出一个选择对话框

【功能说明】该方法与 Alert 方法相似，弹出一个含有指定信息的对话框，但是该方法同 Alert 方法弹出的对话框也有不同之处，调用该方法弹出的对话框中含有一个问号图标和两个按钮，一个按钮的标题为“ 确定 ”，另一个按钮的标题为“ 取消 ”。

该方法的返回值为布尔类型，如果用户单击了“ 确定 ”按钮，则该方法返回 True；否则返回 False。

【基本语法】[blvar=]window.confirm(message)

参数说明如下。

□ message：可选项，字符串表达式，用于指定在弹出的对话框上显示的信息。

□ blvar：可选项，布尔类型，用于存储 Confirm 方法的返回值。

【实例演示】

```
<title>Confirm方法演示</title>
<body>
    <form>
        <input type="button" onClick="testConfirm()" value="关闭窗口">
    </form>
    <script>
            function testConfirm()
            {
                var blvar;
                blvar=confirm("你真的要关闭该窗口吗？");
                if (blvar)
                   window.close();
                else
                   window.alert("你取消了关闭窗口操作!")  ;
            }
    </script>
```

```
</body>
```

运行这段代码后会在窗口中显示一个标题为“ 关闭窗口 ”的按钮，如果单击了该按钮就会弹出一个对话框（如图 16.2 所示），提示是否要关闭窗口。如果选择了“ 确定 ”，则系统会关闭当前窗口；否则关闭窗口的操作将不被执行。

图 16.2　Confirm 方法岩石

16.1.3　prompt 方法：弹出一个供用户输入信息的对话框

【功能说明】该方法弹出对话框，该对话框中含有一个文本框并允许在其中输入信息。在调用该方法时可以设置显示在文本框中的默认值。该方法的返回值为在文本框中输入的内容，如果用户没有在文本框中输入任何内容，则该函数的返回值为 null。

【基本语法】window.prompt(message,defaultvalue)

参数说明如下。

□ message：可选项，字符串表达式，用于指定显示在对话框上的提示信息。

□ defaultvalue：可选项，字符串或某一数字，用于指定显示在文本框中的默认值。

【实例演示】

```
<script>
    var strmsg="请输入你的姓名：";
    var strname="李小虎";
    strname=window.prompt(strmsg,strname);
    if(strname!="" )
    {
        if (window.confirm("你就是大名鼎鼎的"+strname+"吗？"))
            window.alert("见到你真是太荣幸了！！")
    }
</script>
```

运行这段代码弹出对话框，如图 16.3 所示，可以看到这个对话框中含有指定的提示信息和默认值。另外，这段代码中还用到了 Confirm 方法和 Alert 方法，这些代码的功能非常简单，不再赘述。

图 16.3　prompt 方法演示

16.1.4　blur 方法：使 Window 失去焦点

【功能说明】调用该方法后当前 Window 窗口将失去焦点，此时，Window 对象的 onblur 事件将被触发。如果同时打开了多个窗口，则调用该方法后可以使当前窗口成为底层窗口。

【基本语法】window.blur()

【实例演示】下面的代码演示了如何使用 blur 方法使 Window 失去焦点而触发 onblur 事件。

```
<title>blur方法演示</title>
<body onBlur="window.alert('窗口失去了焦点')">
  <form>
```

```
        <input type="button" value="失去焦点" onClick="window.blur()"    />
    </form>
</body>
```

运行这段代码后，会在窗口上出现一个标题为“ 失去焦点 ”的按钮，点击该按钮后，当前窗口将失去焦点，因此，窗口的 onblur 方法会被触发，弹出一个含有“ 窗口失去了焦点 ”字样的对话框，如图 16.4 所示。

图 16.4　blur 方法演示

16.1.5　setInterval 方法：指定每隔多长时间执行指定代码一次

【功能说明】该方法以毫秒为单位设置一个时间间隔，此后，每个指定的时间间隔系统将执行指定的一次代码，直到窗口被关闭或者 clearInterval 方法被调用为止。该方法的返回值为一个整数，该整数为此次调用 setInterval 的标志，调用 clearInterval 方法时使用该整数作为参数可以取消定时执行代码的操作。

【基本语法】window.setInterval(expression,msec[arg1,arg2,…])

参数说明如下。

□ expression：必须项，可执行代码或函数名，用于指定要被反复执行的内容。

□ msec：必选项，常整型数据，以毫秒为单位指定时间间隔。

□ arg1、arg2：可选项，如果 expression 参数指定为某个函数名，可以使用这些选项指定函数参数。

注意：如果 expression 为函数名，则不需要把函数名用引号扩起来，如果为执行语句，则需要使用引号将这些代码扩起来。

【实例演示】下面的代码利用 setInterval 实现了倒计时的功能。

```
<title>setInterval方法演示</title>
<body>
    <form>
        你已经停留<input type="text" name="htime" size="4">秒，
        还可以停留<input type="text" name="time" size="4">秒
    </form>
    <script>
        var inittime=20;
        var havetime=0;
        //设置计时执行代码
        window.setInterval(settime,1000);
        //定义函数，实现时间加减功能
        function settime()
        {
            inittime--;
            havetime++;
            if (inittime==0)
            {
                window.alert("该页面运行的停留时间已经结束，窗口即将关闭");
                window.close();
            }
            document.forms[0].time.value=inittime;
            document.forms[0].htime.value=havetime;
        }
    </script>
</body>
```

这段代码通过 setInterval 函数设置每隔一秒执行函数 settime 一次。该函数每执行一次，

就将用户在该页面的停留时间加 1，而将还可停留的时间减 1。当可以停留的时间为 0 后，就弹出提示对话框，之后将关闭当前窗口。运行这段代码，效果如图 16.5 所示。

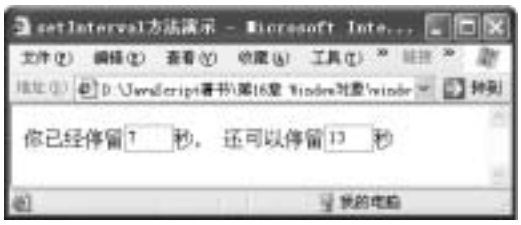

图 16.5　setInterval 函数演示

16.1.6　clearInterval 方法：清除 setInterval 方法产生的作用效果

【功能说明】该方法用于清除 setInterval 函数的作用效果，该方法的参数为 setInterval 函数的返回值，以指定的 setInterval 方法的返回值作为参数，可以清除与之相对应的 setInterval 方法的作用效果。

【基本语法】window.clearInterval(intervalID)

其中，intervalID 为 setInterval 函数的返回值。

【实例演示】下面的代码演示了如何利用 clearInterval 方法清除 setInterval 方法的作用效果。

```
<title>setInterval方法演示</title>
<body >
<form name="frm">
   <input name="txt" border="0" size="0">
   <input type="button" value="停止"    onClick="window.clearInterval(intervalID)">
</form>
<script>
        var intervalID
        intervalID=window.setInterval(setSize,200)
        function setSize()
        {
           document.frm.txt.size++;
        }
</script>
</body>
```

运行这段代码，窗口中的文本框的长度将不断地增加，直到点击了“ 停止 ”按钮。代码中通过 setInterval 方法将文本框的长度每隔 200 毫秒增加一个单位，而单击“ 停止 ”按钮时，将调用 clearInterval 方法结束 setInterval 方法的作用效果，因此，文本框的长度不再增加。执行效果如图 16.6 所示。

图 16.6　clearInterval 方法演示

16.1.7　setTimeout 方法：指定多长时间之后执行指定的代码

【功能说明】该方法同 setInterval 方法不同，setInterval 方法是指定每隔一段时间后就执行指定的代码一次，因此，指定的代码是被反复执行的。而 setTimeout 方法则只在指定的时间间隔之后，只执行指定代码一次，当指定代码被执行后，setTimeout 方法不再发挥作用。

该方法的返回值为一个整数，该整数为此次调用 setTimeout 方法的标志。

【基本语法】window.setTimeout((expression,msec[arg1,arg2,…])

参数说明如下。

□ expression：必项，可执行代码或函数名，指定的内容在规定时间之后，被执行一次。

□ msec：必选项，常整型数据，以毫秒为单位指定时间间隔。

□ arg1、arg2：可选项，如果 expression 参数指定为某个函数名，可以使用这些选项指定函数参数。

【实例演示】下面的代码演示了 setInterval 方法和 setTimeout 方法的不同。

```
<title>setTimeout方法演示</title>
<body >
<form name="frm">
  代码已执行了<input name="txt" value="25" size="3">秒
</form>
<script>
    var timeLen=24;
    window.setInterval("document.frm.txt.value=timeLen--",1000);
    window.setTimeout("window.alert('已经过了5秒')",5*1000);
</script>
</body>
```

这段代码体现了 setInterval 方法和 setTimeout 方法的不同，setInterval 方法每隔 1 秒就执行指定的代码一次，将指定的时间减去 1 秒，而 setTimeout 方法则只在 5 秒后执行一次。运行这段代码可以看到图 16.7 所示的页面效果。

图 16.7　setTimeout 方法演示

16.1.8　clearTimeout 方法：清除 setTimeout 方法的作用效果

【功能说明】在 setTimeout 方法指定的时间间隔之前调用 setTimeout 方法可以取消 setTimeout 方法的作用效果。也就是说，如果在 setTimeout 指定的代码执行之前调用了 learTimeout，则在规定的时间间隔到达以后，指定的代码将不会再被执行。

该方法的参数为 setTimeout 方法的返回值，如果多次使用 setTimeout 方法，则调用 clearTimeout 方法时某个 setTimeout 方法的返回值作为参数，可以清除指定 setTimeout 方法的作用效果。

【基本语法】clearTimeout(intervalID)

其中，intervalID 为 setTimeout 函数的返回值。

16.1.9　close 方法：关闭 Window 窗口

【功能说明】调用该方法后，系统会弹出一个对话框询问是否关闭窗口，如图 16.8 所示。点击“是”窗口将被关闭，点击“否”关闭操作将被取消。如果想直接关闭窗口，而不出现询问对话框，可以在调用 close 方法之前先设置 window 对象的 opener 属性为 null。

【基本语法】window.close()

【实例演示】下面的代码演示了如何直接关闭浏览器窗口。

```
<form name="frm">
   <input type="button" value="关闭窗口" onClick="closeWindow()">
</form>
<script>
      function closeWindow()
      {
          window.opener=null;
          window.close();
      }
</script>
```

执行这段代码会在浏览器窗口上出现一个标题为“ 关闭窗口 ”的按钮，点击该按钮窗口将被直接关闭，而不弹出任何对话框。

图 16.8　询问对话框

16.1.10　focus 方法：使窗口获得焦点

【功能说明】调用该方法之后，window 窗口将成为当前窗口。

【基本语法】window.focus()

16.1.11　moveBy 方法：通过指定偏移量来移动窗口

【功能说明】该方法通过指定窗口左上角的坐标偏移量来移动窗口，即在现在坐标的基础上按指定的值改变窗口的位置。

【基本语法】window.moveBy(ix,iy)

参数说明如下。

□ ix：必选项，整数，窗口左上角横坐标的改变量。

□ iy：必选项，整数，窗口左上角纵坐标的改变量。

注意：这里采用的坐标系统为计算机的默认坐标系统，即屏幕左上角为坐标原点，向右为横坐标正方向，横坐标增加，向下为纵坐标正方向，纵坐标增加。

【实例演示】下面的代码演示了如何利用 moveBy 方法移动窗口。

```
<script>
      var imoved=0;
      var intervalID;
      intervalID=window.setInterval(moveWindow,100);
      function moveWindow()
      {
          //移动窗口
           window.moveBy(1,1);
           imoved++;
           //如果与演示坐标的偏移量达到500则停止移动
           if (imoved>=500)
                window.clearInterval(intervalID);
      }
</script>
```

这段代码运行后，每隔 0.1 秒就将窗口向右和向左分别移动 1 个像素，如果与原始坐标相比，移动的偏移量超过 500，则停止移动窗口。

16.1.12 moveTo 方法：移动窗口到指定的坐标

【功能说明】该方法可以将窗口的左上角移动到指定的坐标处，该方法与 moveBy 方法不同，该方法所指定的数值不是偏移量而是绝对坐标。

【基本语法】window.moveTo(xp,yp)

参数说明如下。

□ xp：必选项，整数，指定的窗口左上角的新的横坐标。

□ yp：必选项，整数，指定的窗口左上角的新的纵坐标。

【实例演示】下面的代码演示了如何使用 moveTo 方法移动窗口。

```
<script>
    var imoved=0;
    intervalID=window.setInterval(moveWindow,100);
    function moveWindow()
    {
        var ipx,ipy;
        //随机产生x y坐标
        ipx=Math.round(Math.random()*800);
        ipy=Math.round(Math.random()*800);
        window.moveTo(ipx,ipy);
        imoved++;
        if (imoved>=50)
            window.clearInterval(intervalID);
    }
</script>
```

这段代码利用随机产生的横坐标和纵坐标来移动窗体，执行这段代码之后，窗口将在屏幕上不停的跳动，直到跳到次数达到 50 次为止。

16.1.13 open 方法：打开一个新的窗口

【功能说明】该方法可以根据指定的属性打开一个新窗口，并可以在该窗口中显示指定 URL 中的内容。该方法的返回值为一个窗口对象，通过该对象可以访问和操作新打开的窗口。

【基本语法】window.open([sURL][,sName][, sFeatures])

参数说明如下。

□ sURL：可选项，字符串表达式，用于指定在新窗口中显示的内容，如果没有指定该项，则在窗口中显示的内容为空。

□ sName：可选项，字符串表达式，用于指定窗口的名称，另外该项也与 HTML 标签<Target>具有相同的作用相同，可以指定窗口的打开位置和类型。表 16.1 列出了发挥这种作用时 sName 的取值及意义说明。

表 16.1　sName 具有特殊意义的可用值及说明

值	说明
_blank	sURL 参数所指定的内容将被显示一个新的、没有名字的窗口中
_media	sURL 参数所指定的内容将被显示在 HTML 文档的媒体工具中，该选项仅在 IE 6.0 及更新版本中可用
_parent	sURL 参数所指定的内容将被显示在当前框架的父框架中，如果当前框架没有父框架，则该选型的作用与_self 相同
_search	sURL 参数所指定的内容将被显示在 IE 的搜索面板中，该选项仅在 IE 5.0 及更新版本中可用
_self	sURL 参数所指定的内容将被显示在当前窗口中
_top	sURL 参数所指定的内容将取代任何将被加载的框架集，如果当前页面中没有

定义任何框架则该选项的作用与_self 参数相同

□ sFeatures：可选项，用于指定新窗口的属性，该选项可以是一个参数也可以是多个参数的组合。关于该参数的可用值及说明请参考表 16.2。

表 16.2　窗口属性的常用设置及说明

属性	说明
fullscreen	指定窗口是否全屏显示，取 yes 表明全屏显示，取 no 不全屏显示
height	以像素为单位指定窗口的高度
width	以像素为单位指定窗口的宽度
left	以像素为单位指定窗口距屏幕左边界的距离
top	以像素为单位指定窗口距屏幕上边界的距离
location	指定窗口中是否显示地址栏，取 yes 表明显示地址栏，取 no 不显示地址栏
resizable	指定窗口是否可改变大小，取 yes 表明窗口可以改变大小，取 no 则不可以改变
scrollbars	指定是否在窗口中显示滚动条，取 yes 显示
status	指定是否在窗口中显示状态栏，取 yes 显示
titlebar	指定是否在窗口中显示标题栏，取 yes 显示
toolbar	指定是否在窗口中显示工具栏，取 yes 显示

【实例演示】

```
<script>
    //在新窗口中打开指定页面
    function openNew()
    {
        window.open("http://www.baidu.com","_blank");
    }
    //在当前窗口中打开指定页面
    function openOnSelf()
    {
         window.open("http://www.baidu.com","_self");
    }
    //打开新窗口并输入一些信息
    function openMsg()
    {
         var newWin;
         newWin=window.open("","公告","height=200,width=200,toolbar=no,
                                    menubar=no,location=no,top=200,left=200");
         newWin.document.write("<h4 align='center'>公告</h4>");
         newWin.document.write("这是一个open方法测试");
    }
</script>
<form>
    <input type="button"   onClick="openOnSelf()" value="在当前窗口打开百度" /><br>
    <input type="button"   onClick="openNew()"      value="在新窗口中打开百度"><br><center>
    <input type="button"   onClick="openMsg()"      value="弹出公告"></center>
</form>
```

这段代码中定义了 3 个函数，分别实现了在新窗口中打开指定页面、在当前窗口中打开指定页面和打开新窗口并输入一些指定信息的功能。运行这段代码可以看到图 16.9 所示的效果。

单击该页面中的第一个按钮，可以将当前窗口中的内容替换为百度首页；单击第二个按钮会在新的窗口中打开百度首页；点击第三个按钮可以弹出图 16.10 所示的窗口。

图 16.9　open 方法演示图

图 16.10　弹出的对话框窗口

16.1.14　navigate 方法：在当前窗口中加载指定页面

【功能说明】该方法可以把指定的页面加载到当前窗口中。

【基本语法】window.navigate(sURL)

其中，sURL 为要加载到当前窗口中的页面的地址。

16.1.15　resizeBy 方法：通过指定窗口右下角坐标的偏移量来缩放窗口

【功能说明】该方法可以通过指定窗口右下角的横坐标和纵坐标的偏移量来移动窗口右下角的位置，而窗口左上角的位置不变，因此起到了改变窗口大小的作用。

【基本语法】window.resizeBy(ix,iy)

语法说明如下。

□ ix：必选项，整型数据，用于指定窗口右下角横坐标的偏移量。

□ iy：必选项，整型数据，用于指定窗口右下角的纵坐标的偏移量。

【实例演示】

```
<script>
    var iresized=0;
    var ixy=-1;
    window.setInterval(resizeWindow,10);
    function resizeWindow()
    {
        window.resizeBy(ixy,ixy);
        iresized++;
        if (iresized>500)
            {
                ixy*=-1;
                iresized=0;
            }
    }
</script>
```

这段代码每隔 10 毫秒就将窗口的大小增加或减小 1 个像素，如果窗口的改变超过 500 个像素，则窗口向相反的方向改变。即：如果窗口增加了 500 个像素，则窗口开始减小；如果窗口减小了 500 个像素，则窗口开始增加。运行这段代码窗口会先减小再增加，如此反复不止。

16.1.16　resizeTo 方法：通过指定窗口右下角的新坐标来改变窗口的大小

【功能说明】该方法同 resizeBy 方法类似，也是通过改变窗口右下角的位置来改变窗口的大。不同的是，resizeBy 指定的是窗口右下角坐标的偏移量，而 resizeTo 方法指定的则是窗口右下角的新的坐标，即窗口右下角直接移动到指定的坐标上。

【基本语法】window.resizeTo(ix,iy)

参数说明如下。

□ ix：必选项，整型数据，用于指定窗口右下角的新的横坐标。

□ iy：必选项，整型数据，用于指定窗口右下角的新的纵坐标。

【实例演示】下面的代码利用 rsizeTo 方法把窗口的大小调整到了屏幕大小。

```
<form>
    <input type="button" onClick="openNewWindow()" value="开始">
</form>
<script>
    var objwin;
    var ih,iw;
    var ihok,iwok;
    var intervalID;
    function openNewWindow()
    {
            ih=iw=100;
            ihok=iwok=0;
            //弹出新的对话框
          objwin=window.open("rsizeTo.html","resizeTo方法测试","height=100,width=100,
          top=0,left=0,resizable=yes");
          //改变窗口的大小，直到窗口达到屏幕大小
            intervalID=objwin.setInterval(resizeWindow,10);
    }
    function resizeWindow()
    {
        ih+=5;
        iw+=5;
        if(ih>screen.height)
            {
                ih=screen.height;
                ihok=1;
            }
        if(iw>screen.width)
            {
                iw=screen.width;
                iwok=1;
            }
        //如果iwok+ihok=2则说明窗口的长和宽均达到屏幕大小
        if(iwok+ihok>=2)
            objwin.clearInterval(intervalID);
        else
                objwin.resizeTo(iw,ih);
    }
</script>
```

运行这段代码后，会在窗口中出现一个“ 开始 ”按钮，单击该按钮，弹出一个长和宽均为 100 像素的窗口，然后，窗口的长和宽开始每隔 10 毫秒增加 5 个像素，直到长和宽均达到屏幕大小为止。

16.1.17 scrollTo 方法：滚动窗口中的内容到新的位置

【功能说明】该方法可以按给定的横坐标和纵坐标的偏移量来滚动显示窗口中的内容。

【基本语法】window.scrollTo(ix,iy)

参数说明如下。

□ ix：必选项，整型数据，以像素为单位指定窗口横坐标的滚动偏移量。

□ iy：必选项，整型数据，以像素为单位指定窗口纵坐标的滚动偏移量。

注意：只有当窗口中的内容无法在窗口中全部显示时，即窗口中出现滚动条时，窗口才可以滚动，否则调用 scrollTo 方法将得不到任何效果。

【实例演示】下面的代码实现了双击自动滚屏的效果。

```
<form onDblClick="toScroll()">
<script>
    var i,intervalID;
    var iw=0;
```

```
        var ih=1;
        document.write("<h4>双击开始滚屏</h4>");
        for(i=1;i<=100;i++)
            {
                document.write(i);
                document.write("<br>");
            }
        function toScroll()
        {
              intervalID=window.setInterval(scrollWindow,10);
        }
        function scrollWindow()
        {
            ih++;
            window.scroll(iw,ih);
        }
    </script>
    </form>
```

这段代码运行时，首先在窗口上输出了 100 行数字，当用户双击窗口中的内容时，程序调用 toScroll 函数开始滚屏，直到窗口最下面的内容显示出来为止。

16.1.18 scrollBy 方法：按给定的偏移量来滚动窗口中的内容

【功能说明】该方法和 scrollTo 方法类似，可以实现窗口的滚动显示。不同的是，scrollTo 方法所指定的参数是窗口要滚动到的新位置，而 scrollBy 所指定的参数则是在现有的基础上所有滚动的新的距离，即 scrollTo 方法指定的滚动距离是滚动距离的总量，而 scrollBy 方法指定的滚动距离是在现有滚动距离基础上的新增量。

【基本语法】window.scrollBy(ix,iy)

参数说明如下。

□ ix：必选项，整型数据，用于指定窗口横坐标的滚动距离增量。

□ iy：必选项，整型数据，用于指定窗口纵坐标的滚动距离增量。

注意：只有当窗口中的内容无法在窗口中全部显示时，即窗口中出现滚动条时，窗口才可以滚动，否则调用 scrollTo 方法将得不到任何效果。

16.1.19 showModalDialog 方法：打开一个模式对话框以显示指定内容

【功能说明】模式对话框是相对非模式对话框来说的，打开一个模式对话框后，该对话框一直具有焦点直到该对话框被关闭，在此期间所有的操作只能针对该模式对话框进行，其他窗口无法获得焦点。showModalDialog 方法可以根据指定的属性打开一个模式对话框并显示指定的内容，另外还可以通过该方法的参数给打开的对话框传送数据。

该方法的返回值为该对话框窗口 returnValue 属性的值。

【基本语法】window.showModalDialog(sURL[,vArguments][,sFeatures])

参数说明如下。

□ sURL：必选项，字符串表达式，用于指定在打开的模式对话框中显示的内容。

□ vArgument：可选项，任意类型的数据，这些数据将被传递给打开的对话框，在对话框中可以使用 dialogArguments 属性来获取传递给对话框的数据。

□ sFeatures：可选项，字符串表达式，用于指定所打开的对话框的属性。关于该选项的部分常用设置即说明请参考表 16.3。

表 16.3　sFeatures 选项的可用值及说明

参数	说明
dialogHeight	整型或浮点型数据，用于指定打开的模式对话框的高度

dialogWidth	整型或浮点型数据，用于指定打开的模式对话框的宽度
dialogLeft	整型或浮点型数据，用于指定打开的模式对话框距屏幕左边界的距离
dialogTop	整型或浮点型数据，用于指定打开的模式对话框距屏幕上边界的距离
center	用于指定对话框是否显示在屏幕中间，默认为居中
edge	用于指定对话框的边框的类型，可用值有两个，sunken 和 raised，默认为 raised
help	用于指定是否在对话框窗口中显示与上下问相关的帮助图标，默认为显示
resizable	用于指定窗口大小是否可调，默认为不可调
scroll	用于指定是否在窗口中显示滚动条，默认为显示
status	用于指定是否在窗口中显示状态栏

说明：dialogHeight、dialogWidth、dialogLeft、dialogTop 的值可以采用 cm，mm，in，pt， pc 或者 px 作为单位，而 center，help，resizable，scroll 和 status 的值为 yes 或 no，表 16.3 中的选项可以组合使用，各选项之间通过“;”隔开。

【实例演示】下面的例子利用 openModalDialog 方法打开了一个模式对话框并在其中输出了一些内容。首先创建一个 HTML 文件并在其中输入如下代码。

```
<title>
    showModalDialog方法测试
</title>
<body>
<script>
    var sFeatures;
     var para;
     //设置传递给对话框的参数
    para="<br>this is a test<br>openModalDialog方法测试" ;
    para=para+"<br>在关闭该窗口以前，其他窗口无法获得焦点";
    sFeatures="center=yes;edge=raised;status=yes;";
    sFeatures=sFeatures+"dialogHeight=250 px;dialogWidth=250 px";
    window.showModalDialog("dialog.html",para,sFeatures);
</script>
</body>
```

然后再创建一个名为“dialog.html”的 HTML 文件并在其中输入如下代码。

```
<p>openModalDialog方法测试。</p>
<p> 获取的参数为：
<script>
    document.write(window.dialogArguments);
</script>
</p>
```

运行第一段代码，窗口中弹出一个对话框，如图 16.11 所示。

图 16.11　showModalDialog 方法演示

可以看到指定的数据成功的传递给了打开的模式对话框，而且在对话框中可以很容易地获取并显示这些数据。

16.1.20 showModalessDialog 方法：打开一个非模式对话框并显示指定内容

【功能说明】非模式对话框与模式对话框不同，在打开非模式对话框时还可以对其他窗口进行操作。showModalessDialog 方法的作用就是创建一个非模式对话框并在其中显示指定的页面内容。

该方法的返回值与 showModalDialog 方法也不同，该方法的返回值为打开的窗口对象。

【基本语法】window.showModalessDialog(sURL[,vArguments][,sFeatures])

参数说明如下。

□ sURL：必选项，字符串表达式，用于指定在打开的非模式对话框中显示的内容。

□ vArgument：可选项，任意类型的数据，这些数据将被传递给打开的对话框，在对话框中可以使用 dialogArguments 属性来获取传递给对话框的数据。

□ sFeatures：可选项，字符串表达式，用于指定所打开的对话框的属性。关于该选项的部分常用设置即说明请参考表 16.3。

【实例演示】

```
<script>
    var sFeatures;
      var para;
      var newWin;
      para="showModalessDialog方法演示" ;
    para+="<br>打开该窗口后还可操作其他窗口<br>";
    para+="<center><input type='button' value='关闭该窗口' onclick='window.opener=null;
    window.close();'</center>"
      sFeatures="center=yes;edge=raised;status=yes;resizable=yes;";
      sFeatures=sFeatures+"dialogHeight=250 px;dialogWidth=250 px";
      newWin=window.showModelessDialog("dialog.html","",sFeatures);
      newWin.document.write(para);
</script>
```

这段代码通过 showModalessDialog 方法打开了一个非模式对话框，然后通过对该对话框窗口的引用向对话框中写入了一些内容。运行这段代码可以看到图 16.12 所示的效果。

图 16.12　showModalessDialog 方法演示

16.2 Window 对象的属性

16.2.1 closed 属性：判断引用的窗口是否已经关闭

【功能说明】该属性为只读属性，其返回值为布尔子类型，如果引用的窗口已经关闭，则该属性返回 True，否则，返回 False。

【基本语法】window.closed

【实例演示】下面的代码计算了弹出窗口的存在时间。

```
<script>
    var newWin;
    var intervalID;
    var timeLen=0;
    function getTimeLen()
    {
        //如果窗口已经关闭
        if (newWin.closed)
          {
            window.clearInterval(intervalID);
             window.alert("窗口打开了"+timeLen+"秒");
          }
         else
            timeLen++;
    }
    function openNew()
    {
        newWin=window.open("","公告","height=200,width=200,toolbar=no,
                                menubar=no,status=yes,location=no,top=200,left=200");
        newWin.document.write("<h4 align='center'>公告</h4>");
        newWin.document.write("这是一个closed属性测试");
        newWin.document.write("<center><input type='button' value='关闭窗口' onclick='window.opner=null;
                                window.close();'</center>");
        intervalID=window.setInterval(getTimeLen,1000);
    }
</script>
<form><input type="button" onclick="openNew()" value="打开窗口"></form>
```

这段代码利用 setInterval 方法每隔一秒钟调用 closed 属性判断一下窗口是否已经关闭。如果窗口已经关闭，就弹出对话框显示窗口一共存在了多长时间（如图 16.13 所示）；否则就将变量 timeLen 加 1，以记录窗口存在的时间。

图 16.13　closed 属性演示

16.2.2　defaultStatus 属性：设置或返回窗口的缺省状态信息

【功能说明】该属性为可读写属性，利用该属性可以设置或读取浏览器窗口的默认状态信息。如果没有设置该属性，则打开一个窗口后，其默认状态信息为 “ 完成 ”。

【基本语法】wondow.defaultStatus[=smessage]

其中，smessage 就是要设置的状态信息。

【实例演示】下面的代码演示了如何使用 defaultStatus 属性设置窗口的默认状态信息。

```
<script>
    window.defaultStatus="defaultStatus属性测试"
</script>
```

运行这段代码，效果如图 16.14 所示。从图中可以看到状态栏上所设置的信息。

图 16.14　defaultStatus 属性演示

16.2.3 dialogArguments 属性：获取传递给模式对话框的数据

【功能说明】在使用 showModalDialog 方法创建模式对话框时，可以使用该方法的第二个参数给打开的模式对话框传递数据。在模式对话框所显示的页面中可以模式对话框的 dialogArguments 属性来获取传递给对话框的数据。

【基本语法】[data=]window. dialogArguments

其中，data 可以为任意类型的变量，用来接收 dialogArguments 属性的值。关于该属性的演示实例请参考 showModalDialog 方法，在此不再赘述。

16.2.4 dialogHeight、dialogWidth 属性：设置或返回模式对话框的高度、宽度

【功能说明】利用这两个属性可以设置或读取所打开的模式对话框的高度和宽度。设置时可采用 cm，mm，in，pt，pc 或者 px 为单位，默认为 px。

【基本语法】window.dialogHeight[=nheight]　window.dialogWidth[=nwidth]

其中，nheight 和 nwidth 均为整型数据，用于设置对话框的高度和宽度。

注意：只有使用 showModalDialog 和 showModalessDialog 方法打开的窗口才可以使用这两个属性。

dialogHeight 属性和 dialogWidth 属性一般在 showModalDialog 和 showModalessDialog 方法的第三个参数（sFeatures）中使用，具体的演示实例请参考第 16.19 小节和第 16.20 小节的演示实例，在此不再赘述。

16.2.5 dialogLeft、dialogTop 属性：设置或返回对话框的位置

【功能说明】dialogLeft 属性用于设置对话框左侧距屏幕左边界的距离，也就是对话框左上角的横坐标。dialogTop 属性用于设置对话框上侧距屏幕上边界的距离，也就是对话框左上角的纵坐标。这两个属性和 dialogHeight 属性与 dialogWidth 属性相同，只能作用于使用 showModalDialog 或者 showModalessDialog 方法打开的模式对话框或非模式对话框。

【基本语法】window.dialogLeft[=nleft]　window.dialogTop[=ntop]

其中，nleft 和 ntop 均为整型数据，可用的单位与 dialogHeight 属性相同，默认为 px。

16.2.6 opener 属性：设置返回对打开当前窗口的副窗口的引用

【功能说明】如果当前窗口是其他窗口使用 open 方法打开的窗口，那么在当前窗口中使用 opener 属性可以返回对当前窗口的创建者的引用，因此，通过 opener 属性可以访问创建当前窗口的副窗口中的数据。

【基本语法】[objWindow=]window.opener

其中，objWindow 为一个 widow 对象，是对打开当前窗口的副窗口对象的引用。下面的例子演示了 opener 属性的作用和用法。首先创建一个 HTML 文件并在其中输入如下代码。

【实例演示】

```
<script>
    var str="width=200,height=200,toolbar=no,status=no";
</script>
<form action="search.html">
    姓名<input type="text" size="10" name="name"><br>
    年龄<input type="text" size="4" name="age"><br>
    住址<input type="text" name="address"><br>
```

```
    <input type="button" value="提交" onClick="window.open('opener.html','opener属性演示',str)">
</form>
```

然后再创建一个名为“ search.html ”的文件并在其中输入如下代码。

【实例演示】

```
<script>
    document.write("请确认您提交的信息:<br>");
    with(window.opener.document.forms[0])
    {
        window.document.write("<li>姓名:"+name.value+"</li>");
        window.document.write("<li>年龄:"+age.value+"</li>");
        window.document.write("<li>地址:"+address.value+"</li>");
         window.document.write("<input type='button' value='关闭窗口' onclick='closeWindow()'>");
    }
    function closeWindow()
    {
        window.opener=null;
         window.close();
    }
</script>
```

这个实例中第一段代码创建了一个 HTML 表单，在其中输入数据（如图 16.15 所示）并单击“ 提交 ”按钮，就会弹出一个新窗口以显示用户输入的数据，如图 16.16 所示。在弹出的窗口中使用 opener 属性来引用原窗口而读取到原窗口表单中的数据。

图 16.15　在表单中输入数据

图 16.16　弹出窗口

从本例中可以看到，通过 opener 属性可以在打开的窗口中方便地访问打开该窗口的副窗口中的数据。如果存在多级打开关系，比如：在窗口 A 中打开了窗口 B，而在窗口 B 中又打开了窗口 C，那么就可以使用 C.opener.opener 来引用窗口 A，并可以访问窗口 A 中的数据。

16.3　Window 对象的子对象

Window 对象是 DOM 对象模型中的顶层对象，其他对象都是 Window 的子对象，本节只介绍几个简单的子对象，其他复杂的对象将单独作为一章来介绍。

16.3.1　screen 对象：获取计算机屏幕的一些属性

screen 对象有很多属性，利用这些可以获得关于计算机屏幕的一些信息，比如长度、宽度等，这里只介绍几个比较常用的属性。

1．availHeight 属性

【功能说明】该属性用于获取计算机屏幕工作区域的高度，所谓的工作区域就是指计算机屏幕上除了工具条以外的其他区域，利用 availHeight 属性可以获取之一区域的高度。

【基本语法】[iheight=]screen.availHeight

其中，iheight 为整型变量，单位为 px（像素），用于接收该属性的值。

2．availWidth 属性

【功能说明】该属性以像素为单位返回计算机屏幕工作区域的宽度。该属性和 availHeight 属性一样都是只读属性，不能改变它们的值。

【基本语法】[iwidth=]screen.availWidth

其中，iwidth 为整型变量，用于接收 availWidth 属性的返回值。下面的代码演示了如何利用 availHeight 属性和 availWidth 属性获取计算机屏幕工作区域的尺寸。

【实例演示】

```
<script>
    var iheight=100;
     var iwidth=100;
    var intervalID;
    //移动窗口至计算机屏幕的右上角并调整窗口大小
    window.moveTo(0,0);
    window.resizeTo(iwidth,iheight);
    intervalID=window.setInterval(loadWindow,1);
    function loadWindow()
    {
        /*  首先增加窗口的高度至屏幕工作区域高度
            然后再增加窗口的宽度至屏幕工作区域的宽度
            窗口的高度和宽度都达到屏幕工作区域的尺寸时，停止*/
        if(iheight<screen.availHeight)
            iheight+=3;
        else if(iwidth<screen.availWidth)
            iwidth+=3;
        else
             window.clearInterval(intervalID);
        window.resizeTo(iwidth,iheight);
    }
</script>
```

这段代码利用 screen 对象的 availHeight 属性和 availWidth 属性获取了计算机屏幕工作区域的尺寸，然后逐渐增加窗口的高度至计算机屏幕工作区域的高度，再增加窗口的宽度至计算机屏幕工作区域的宽度。

3．height、width 属性

【功能说明】这两个属性以像素为单位返回了计算屏幕的垂直分辨率和水平分辨率，因此，这两个属性的返回值均为整型数据。

【基本语法】screen.height　screen.width

【实例演示】

```
<script>
    var iheight;
    var iwidth;
    iheight=screen.height;
    iwidth=screen.width;
    window.document.write("计算机的当前分辨率为:"+iheight+"×"+iwidth);
</script>
```

这段代码利用 height 属性和 width 属性获取了当前计算机的分辨率。运行这段代码后，可以看到效果如图 16.17 所示。

图 16.17　height 和 width 属性演示

16.3.2 location 对象：设置或获取当前 URL 的信息

使用 location 对象可以设置或返回 URL 中的一些信息，一个完整的 URL 地址的格式为：

协议://主机:端口/路径名称#hash 标识?搜索条件

其中，协议是 URL 的起始部分，用于指定该 URL 地址所采用的通信协议，比如 http、ftp 等；主机是指该 URL 所对应的服务器的名称；端口用于指定服务器用于通信的端口号，与主机名之间使用冒号隔开；路径名称是指该 URL 所对应的网页文件在服务器上的虚拟路径；如果页面中含有锚点连接，可以使用 hash 标志指定页面中的锚点标志，该标志以“ # ”开头；搜索条件是指 URL 中所含有的查询条件，该查询条件以“ ? ”开头，以“ 变量名称=值 ”的形式出现，多个查询条件之间使用连接符“ & ”连接。比如：http://upc.edu.cn:8080/wwwroot/index.html#topicID?id=3876。

利用 location 对象可以方便地设置或获取 URL 中的各种信息，本节将详细介绍 location 对象的一些常用属性和方法。

1．hash 属性

【功能说明】设置或获取 URL 中的锚点名称，如果 Web 页面中使用的锚点连接，通过设置 location 对象的 hash 属性可以方便的跳转到页面中的不同部分。

【基本语法】location.hash

下面的代码演示了如何在网页中使用 location 对象的 hash 标志快速定位页面中的内容。

【实例演示】

```
<script>
      function getAnchor(str)
      {
        window.location.hash=str;
      }
</script>
<body>
  //创建锚点链接，快速定位网页内容
  <a href="javascript:getAnchor('jueju');">杜甫 绝句</a>
  <a href="javascript:getAnchor('yijianmei');">李清照 一剪梅</a>
  <a href="javascript:getAnchor('hanghelou');">崔颢 黄鹤楼</a>
  //下面定义了3个锚点
  <a name="jueju"><center>绝句</center>
  <!--古诗内容省略，见源代码-->
  </a><br><br><br><br><br>
  <a name="yijianmei"><center>一剪梅</center>
  <!--古诗内容省略，见源代码-->
  </a><br><br><br><br><br>
  <a name="hanghelou"><center>黄鹤楼</center>
  <!--古诗内容省略，见源代码-->
  </a>
</body>
```

这段代码在一个网页文件中创建了 3 个锚点链接，单击其中任一个链接，程序就会调用 getAnchor 函数来设置 location 对象的 hash 属性为相应的值，以使页面滚动到指定的内容上。运行这段代码，将窗口缩小并单击链接“ 崔颢 黄鹤楼 ”则可以看到图 16.18 所示的效果。

图 16.18　hash 属性演示

2．host 属性

【功能说明】设置或返回 URL 地址中主机的名称和端口号。如果 URL 中没有显示的指定端口号，则 host 属性的返回值和 hostname 属性的返回值相同。

【基本语法】location.host

3．hostname 属性

【功能说明】设置或返回 location 对象或 URL 地址中的主机名称或者主机的 IP 地址。对大多 Web 站点服务器来说，该属性的返回值还包含域名和 3w 标志。

【基本语法】location.hostname

4．port 属性

【功能说明】设置或返回 URL 地址中的服务器端口号。如果 URL 中没有指定通信端口号，则该属性的返回值为空字符串。

【基本语法】location.port

5．pathname 属性

【功能说明】设置或返回 URL 所对应的网页文件的虚拟路径，其中包括网页文件的文件名。

【基本语法】location.pathname

6．protocol 属性

【功能说明】设置或返回 URl 中所包含的通信协议，其中包括“：”，如 http:、ftp:等。

【基本语法】location.protocol

【实例演示】下面的代码演示了如何利用上面几个属性获取 URL 中的相应信息。

```
<table align="center" border="1" bordercolor="#009900" style="border-collapse:collapse">
    <script>
        with(document)
        {
            writeTable("属性","属性值");
             writeTable("URL",location.href);
             writeTable("host",location.host);
             writeTable("hostname",location.hostname);
             writeTable("port",location.port);
             writeTable("pathname",location.pathname);
             writeTable("protocol",location.protocol);
        }
        function writeTable(str1,str2)
        {
            with (document)
            {
                write("<tr>");
                 write("<td>");write(str1);write("</td>");
                 write("<td>");write(str2);write("</td>");
                 write("</tr>");
            }

        }
    </script>
</table>
```

这段代码通过自定义函数以表格的形式输出了 location 对象的部分属性和属性值，运行这段代码可以看到图 16.19 所示的页面效果。

7．href 属性

【功能说明】该属性为 location 对象的默认属性，是 location 对象中最常用的属性。利用该属性可以设置或返回整个 URl 字符串，通过重新设置 location 对象的 URL 地址，可以使窗口中的内容跳转到指定的 Web 页面。

【基本语法】location.href[=surl]

其中，surl 是要跳转到的 URL 地址。

注意：href 属性返回的 URL 地址是经过编码以后的字符串，比如空格被显示为%20，要想得到编码前的 URL 字符串信息，可以使用 unescape()函数进行处理。

【实例演示】下面的代码演示了如何通过设置 location 对象的 href 属性来跳转到指定的页面。

```
<body>
选择页面
<select onChange="window.location.href=this.options[this.selectedIndex].value">
      <option value="http://www.baidu.com">百度</option>
       <option value="http://www.google.com">谷歌</option>
       <option value="http://www.hao123.com">hao123</option>
</select>
</body>
```

运行这段代码可以看到图 16.20 所示的页面，从列表框中选择一个页面，程序就会加载相应的页面到当前窗口中。

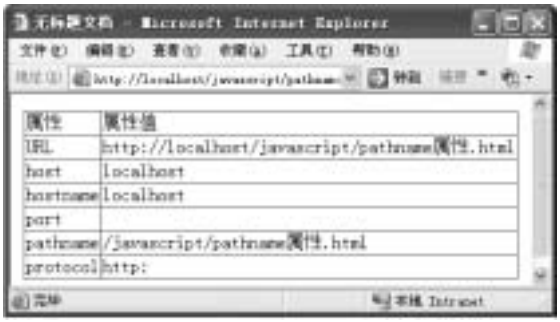

图 16.19　location 对象属性的使用演示

图 16.20　href 属性演示

8．search 属性

【功能说明】设置或返回 URL 地址中的查询信息，即 URL 中的问号及问号以后的信息。如果 HTML 表单中的数据采用 get 方法进行传输（默认），则表单中的数据信息将以查询条件的形式传送给处理页面。因此可以使用 location 对象的 search 属性来获取这些数据并进行处理。

【基本语法】[sSearch=]location.search

其中，sSearch 为返回的查询条件字符串。

【实例演示】下面的例子演示了如何获取并处理 URL 中的查询字符串。首先创建一个 HTML 表单，主要代码如下。

```
<form action="search.html">
    姓名<input type="text" size="10" name="name"><br>
    年龄<input type="text" size="4" name="age"><br>
    住址<input type="text" name="address"><br>
    <input type="submit" value="提交">
</form>
```

这个 HTML 表单中指定了使用“ search.html ”来处理表单中的数据，该文件中的主要代码如下所示。

```
<script>
        var arrSearch=new Array();
        //调用函数处理查询条件信息
        analysisSearch();
        writeSearchInfor();
        //分析location.search属性的值，从中分离出变量和值
        function analysisSearch()
          {
            var sSearch=unescape(location.search);
            sSearch=sSearch.substr(1);
            arrtemp=sSearch.split("&");
            for(var i=0;i<arrtemp.length;i++)
            {
              temp=arrtemp[i].split("=");
```

```
                arrSearch[temp[0]]=temp[1];
            }
        }
        //输出分析结果
        function writeSearchInfor()
        {
            document.write(arrSearch["name"]+"你好,下面是您的个人信息：<br>");
            for(var sitem in arrSearch)
            {
                var str;
                str="<li>"+sitem+":"+arrSearch[sitem]+"</li>"
                document.write(str);
            }
        }

</script>
```

这段代码中定义了一个函数 anlysisSearch，该函数利用 String 对象的一些方法函数把 location 对象的 search 属性返回的查询条件字符串分割并存储在了数组 arrSearch 中。函数 writeSearchInfor 则实现了输出分析结果的功能。

首先运行含有 HTML 表单的文件并在表单中填写数据，如图 16.21 所示，单击“ 提交 ”按钮后，程序会调用“ search.html 处理 ”表单中的数据并输入分析结果，如图 16.22 所示。

图 16.21　HTML 表单中输入的数据

图 16.22　分析结果

9．assign 方法

【功能说明】该方法的作用与 href 属性的作用效果相同，可以将当前窗口中的显示内容跳转到指定的 URL 地址。

【基本语法】location.assign(sURL)

其中，sURL 为要跳转到的 URL 地址。

【实例演示】将 href 属性的演示实例的代码修改如下，可以实现相同的效果。

```
<body>
选择页面
<select onChange="window.location.assign(this.options[this.selectedIndex].value)">
    <option value="http://www.baidu.com">百度</option>
     <option value="http://www.google.com">谷歌</option>
     <option value="http://www.hao123.com">hao123</option>
</select>
</body>
```

10．reload 方法

【功能说明】该方法的作用与浏览器工具栏上的刷新按钮相似，可以重新加载当前页面至浏览器窗口，该方法的功能更加强大。

【基本语法】location.reload([breload])

其中，breload 为布尔型变量或常量，为 True 表明要从服务器上重新加载当前页面；为 False 时表明从缓存中重新加载当前页面。

【实例演示】下面的代码演示了 reload 方法和使用浏览器刷新的区别。

```
<form>
文本框<input type="text" >
<input type="checkbox" value="test">复选框<br>
列表框<select>
    <option value="http://www.baidu.com">百度</option>
```

```
        <option value="http://www.google.com">谷歌</option>
        <option value="http://www.hao123.com">hao123</option>
</select><br>
<input type="button" value="从服务器刷新" onClick="window.location.reload(true);">
<input type="button" value="从缓存中刷新" onClick="window.location.reload(false);">
</form>
```

运行这段代码并在其中输入一些数据如图 16.23 所示，单击浏览器上的刷新按钮以后，表单中的数据将继续存在，而单击表单中的刷新按钮之后表单中的数据将被清空。

11．replace 方法

【功能说明】使用指定的页面来替换当前窗口中的文档内容，使用该方法后当前页面也会被从 history 对象中清除掉，因此当前页面不会出现在浏览器的历史记录中，使用浏览器的“ 前进 ”或者“ 后退 ”将无法再次查看本页面。

【基本语法】location.replace(sURL)

其中，sURL 是要用来替换当前窗口中文档内容的页面的 URL 地址。。

【实例演示】下面的代码演示了 replace 方法的用法和作用

```
<script>
    function toReplace(sURL)
    {
        window.location.replace(sURL);
    }
</script>
<form>
    <input type="button" value="替换" onClick="toReplace('16.3.2    location对象_hash.html')">
</form>
```

运行这段代码后窗口会出现一个“ 替换 ”按钮，单击该按钮后程序会调用“ 16.3.2 location 对象_hash.html ”页面来替换当前页面，如图 16.24 所示，可以看到浏览器窗口中的“ 后退 ”按钮处于不可用状态，因此无法使用“ 后退 ”按钮后退到替换前的页面中。

图 16.23　reload 方法演示

图 16.24　replace 方法演示

16.3.3　history 对象：访问最近所访问的 URL 的列表

在浏览 Web 页面时，浏览器会将最近访问过的页面的 URL 地址保存在一个地址中，使用 history 对象可以访问这一列表。history 对象共有 4 个属性和 3 个方法依次介绍如下。

1．current 属性

【功能说明】该属性为只读属性，用于返回在历史记录列表中当前所访问的历史页面的 URL 地址。

【基本语法】history.current

说明：由于用户的历史页面中往往含有敏感信息，如果所有的 Web 页面都可以使用 history 对象获取到用户的历史访问记录，就会导致用户的隐私或敏感数据被他人窥探，因此，只有具有签名脚本的网页才可以使用 history 对象的 current、next 和 previous 属性来获取历史页面的 URL 地址。

2．next 属性

【功能说明】该属性为只读属性，用于返回在历史记录列表中当前所访问的历史页面的

下一条记录的 URL 地址。

【基本语法】history.next

3．previous 属性

【功能说明】该属性为只读属性，用于返回在历史记录列表中当前所访问的历史页面的上一条记录的 URL 地址。

【基本语法】history.previous

4．length 属性

【功能说明】只读属性，返回历史记录列表中的记录条数。

【基本语法】[nCount=]history.length

其中，nCount 为整型数据，用于存储 length 属性返回的历史列表中的记录数目。

【实例演示】下面的代码演示了如何使用 length 属性得到当前窗口中的历史记录的数目。

```
<script>
    var nCount=window.history.length;
    document.write("您曾使用该窗口访问过"+nCount+"个页面");
</script>
```

运行这段代码可以看到图 16.25 所示的页面。

5．back、forward 方法

【功能说明】该两个方法的作用分别与浏览器工具栏上的“ 后退 ”和“ 前进 ”按钮的作用相同，通过 back 方法可以返回到历史记录中前一条记录所对应的页面；而使用 forward 方法则可以跳转到历史记录中下一条记录所对应的网页中。

【基本语法】history.back()

history.forward()

【实例演示】

```
<body>
  <a href="javascript:window.history.back();">前进</a>
  <a href="javascript:window.history.forward();">后退</a>
</body>
```

这段代码在页面上创建了“ 前进 ”和“ 后退 ”的超级链接，如图 16.26 所示，通过这两个链接可以跳转到历史记录中的上一页和下一页。

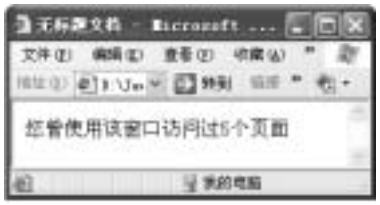

图 16.25　length 属性演示

图 16.26　back 和 forward 方法演示

6．go 方法

【功能说明】从历史列表中加载指定的页面到当前窗口。如果指定的页面超出了历史记录的范围，则历史记录中的最后一页或者最前一页将被显示在窗口当中。

【基本语法】history.go(vloation)

其中，vlocation 为必选参数，整数或者字符串，用于从历史记录中调用页面。如果为整数则调用历史记录列表中相对为当前记录的页面 ;如果为字符串则为历史记录中某一记录所对应的 URL 地址。因此，可以使用 history.go(-1)和 history.go(1)实现 history.back 和 history.forward 的作用。

【实例演示】使用下面的代码同样可以实现前进和后退的功能。

```
<body>
    <a href="javascript:window.history.go(-1);">前进</a>
    <a href="javascript:window.history.go(1);">后退</a>
</body>
```