CHAPTER 4

第 4 章 DHTML 动态网页制作

学习要点
（1）掌握 JavaScript 语言的语法结构和流程控制。 （2）掌握 JavaScript 语言的事件和对象编程方法。 （3）学会使用 JavaScript 内置对象编程。 （4）了解 HTML DOM 主要对象，jQuery 选择器，掌握 HTML DOM、jQuery 编程技术。

上一章介绍了 HTML 常用的标记和 CSS 技术，可以创建出具有复杂格式的网页，但这些页面是静态的，用户只能静态地查看所显示的内容，而无法和网页进行交互。一旦静态页面上需要改变某个内容，则必须在原来的页面上修改，甚至会发生很大的布局变化。这往往会给设计人员带来许多不便，尤其是对大型网站，其信息呈现的方式不断变化，如果采用静态页面技术，对设计人员来说可能是一种灾难。

每一个 HTML 标记或者称为 HTML 元素（有时也称 HTML 标签）都可看成一个在浏览器中显示的对象。可以利用脚本语言例如 JavaScript 来操控这些对象，使它们的行为和外观在浏览器中动态改变。这就是所谓的 DHTML（Dynamic HTML）技术。

DHTML 并不是一门新的语言，它只是 HTML、CSS 和一种脚本程序的集成。其中：

- HTML 元素，也即页面中的各种标记对象，它们是被动态操纵的内容；
- CSS 属性，也可被动态操纵，从而获得动态的格式效果；
- 脚本程序（如 JavaScript，VBScript），它实际操纵 Web 页上的 HTML 和 CSS。

脚本程序区分为客户端脚本程序（通过客户端浏览器解释执行）和服务器端脚本程序（在服务器端解释或编译后执行，例如 ASP/ASP.NET、JSP、PHP 等）。客户端脚本比服务器端脚本程序在执行任务上有明显的优越性，由于客户端脚本程序随网页同时下载到客户机上浏览器缓存中，通过浏览器解释执

行，因此在对网页进行验证或响应用户动作时无须使用网络与 Web 服务器进行通信，从而降低了网络的传输量和 Web 服务器的负荷，改善了系统的整体性能。但客户端脚本程序并不能解决所有问题，例如数据库操作、访问计数器等，必须使用服务器端脚本程序来实现。本章仅讨论客户端脚本程序。

DHTML 使网页设计者可以动态操纵网页上的所有元素。利用 DHTML，网页设计者可以动态地隐藏或显示内容、修改样式定义、激活元素以及为元素定位。此外，网页设计者还可利用 DHTML 在网页上显示外部信息，方法是将元素捆绑到外部数据源（如文件和数据库）上。所有这些功能均可用浏览器完成而无须请求 Web 服务器，同时也无须重新装载网页。

某些浏览器对微软公司推出的脚本语言 VBScript 支持不力，微软公司又推出了脚本语言 JScript，它和 Netscape 公司推出的 JavaScript 语言在用法上几乎完全相同。本章着重介绍 JavaScript 语言及其相关动态网页制作技术。

4.1　JavaScript 编程技术及实例

4.1.1　JavaScript 语言简述

JavaScript 是一种嵌入 HTML 文件中的脚本语言，它是基于对象驱动的，能对鼠标单击、表单输入、页面浏览等用户事件做出反应并进行处理。JavaScript 具有以下特点。

1．简单性

JavaScript 是简化的编程语言，不像高级语言有严格的使用限制，使用简洁灵活。例如在 JavaScript 中可直接使用变量，不必事先声明，变量类型规定也不十分严格。

2．基于对象

JavaScript 是一种基于对象（object-based）的语言，允许用户自定义对象，同时浏览器还提供大量的内建对象，可以将浏览器中不同的元素作为对象处理，体现了面向对象编程的思想。但 JavaScript 并不完全面向对象，不支持类和继承。

3．可移植性

JavaScript 可在大多数浏览器上不经修改直接运行。

4．动态性

JavaScript 是 DHTML 的重要组成部分，是设计交互式动态特别是客户端动态页面的重要工具。

5．安全性

JavaScript 是一种安全性语言，它不允许访问本地的硬盘，并不能将数据存入到服务器上，不允许对网络文档进行修改和删除，只能通过浏览器实现信息浏览或动态交互，从而有效地防止数据的丢失。

JavaScript 与 Java 在命名、结构和语言上都很相似，两者存在重要的差别。

（1）Java 是 Sun 公司推出的新一代面向对象的程序设计语言，支持类和继承，主要应用于网络编程；JavaScript 只是基于对象的，主要用于 Web 页面编写脚本，是 Netscape 公司的产品。

（2）Java 程序编译后以类的形式存放在服务器上，由浏览器下载用 Java 虚拟机去执行它。JavaScript 源代码嵌入 HTML 文件中，使用时由浏览器对它进行识别、解释并执行。

（3）Java 采用强变量检查，即所有变量在编译之前必须声明。JavaScript 中变量声明，采用弱变量，在使用前不需作声明，而是解释器在运行时检查其数据类型。

（4）Java 程序可单独执行，而 JavaScript 程序只能嵌入 HTML 中，不能单独执行。

（5）Java 程序的编写、编译需要专门的开发工具，如 JDK（Java Development Kit）、Visual J++等；而 JavaScript 程序只是作为网页的一部分嵌入 HTML 中，编写 JavaScript 程序只要用一般的文本编辑器即可。

4.1.2 JavaScript 编程基础

1．将 JavaScript 程序嵌入 HTML 文件的方法

（1）在 HTML 文件中使用<script>、</script>标记加入 JavaScript 语句，可位于 HTML 文件的任何位置。最好是将所有脚本程序放在 HEAD 标记内，以确保容易维护。在 Script 标记之间加上“<!--”和“//-->”表示如果浏览器不支持 JavaScript 语言，这段代码不执行。

【例 4.1】 HTML 文件中使用脚本语言示例 1。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>HTML 中如何使用 script 语言--设置收藏夹实例</title>
</head>
<script language="javascript">
<!--
alert('Hello'); //显示消息对话框
 function Add_Favorite(url,title)
{
    window.external.AddFavorite(url,title); //放到收藏夹
 }
-->
</script>
<body>
<A HREF="javascript:Add_Favorite('http://cqu.edu.cn','重庆大学');">收藏本站
</A>
</body></html>
```

代码可以放在函数中，也可以不放在函数中。不放在函数中的代码在浏览器加载 HTML 页面后还没有呈现 HTML 显示效果前就执行一次，以后不再执行，除非你重新加载页面。而函数则可根据用户需要在页面中多次调用，完成多次执行操作，可实现页面级的代码共享。例如上例中 alert 仅执行一次。HTML 页面中可有多个“<script></script>”程序段，程序段的前后关系以及程序段与 HTML 标记之前的前后关系应有逻辑关系。下例中第

一程序段不能放到<A>标记后，想想为什么？

```
<html><head><title>脚本位置的问题</title></head>
<script type="text/javascript">
var Num=9;  //此处定义的变量 Num 将直接传递给单击超链接的事件处理函数 ClacMe
var Str="How much ? "; //此处的 Str 将传递给后面的脚本程序使用
 </script>
<body>
<A  id="myAlink" HREF="#" onclick="CalcMe(Num);">计算</A>
</body>
<script type="text/javascript">
function CalcMe(n)
   { n=Num*90; alert(Str+n) }   //最终显示"How much ? 810"
</script>
</html>
```

W3C 建议在 HTML 4.0 版本中指定脚本语言用 type 属性替代 language 属性，例如上例中<script type="text/javascript"> 。

（2）将 JavaScript 程序以扩展名".js"单独存放，再使用<script src=*.js>嵌入到 HTML 文件中，有利于实现代码共享。

【例 4.2】 HTML 文件中使用脚本语言示例 2。

```
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>HTML 中如何使用 script 语言--设置主页实例</title>
</head>
<script src="sethomepage.js" language="javascript"></script>
<body>
<A  id="myAlink" HREF="#" onclick="Set_HomePage();">设为主页</A>
</body>
</html>

sethomepage.js 文件内容:
function Set_HomePage()
   {
        myAlink.style.behavior='url(#default#homepage)';
        myAlink.setHomePage('http://www.cqu.edu.cn/');
        return false;
   }
```

（3）直接在 HTML 标记内添加脚本。例如例 4.2 可改成：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>HTML 中如何使用 script 语言--设置主页实例</title> </head>
```

```
<body>
```

```
<A HREF="#" onclick="javascript:this.style.behavior='url(#default#homepage)';
 this.setHomePage('http://www.cqu.edu.cn/'); return false;;"> 设为主页</A>
</body></html>
```

其中 this 指代当前标记元素对象。当实现功能较为简单，而且代码不需要共享时，可采用此种方法书写脚本程序。

2．数据类型

JavaScript 有三种数据类型：字符型、数值型和布尔型。

1）字符型

字符串数据类型用来表示 JavaScript 中的文本。脚本中的字符串文本放在一对匹配的单引号或双引号中。字符串中可以包含双引号，该双引号两边需加单引号，例如'4"5'，也可以包含单引号，该单引号两边需加双引号，例如"1'5"。

【例 4.3】 下面是字符串的示例。

```
"Happy am I; from care I'm free!"
'"Avast, ye lubbers!" roared the technician.'
"42"
'c'
```

JavaScript 中没有表示单个字符的类型（如 C++的 char）。要表示 JavaScript 中的单个字符，应创建一个只包含一个字符的字符串。包含 0 个字符（""）的字符串是空（0 长度）字符串。

2）数值型

在 JavaScript 中，整数和浮点值没有差别；JavaScript 数值可以是其中任意一种（JavaScript 内部将所有的数值表示为浮点值）。

整型值可以是正整数，负整数和 0。可以用十进制，八进制和十六进制来表示。在 JavaScript 中数字大多是用十进制表示的。浮点值为带小数部分的数，也可以用科学计数法来表示。相关示例如表 4-1 所示。

表 4-1　JavaScript 中数据类型示例

数　　字	描　　述	等价十进制数
3.45e2	浮点数	345
42	整数	42
378	十进制整数	378
0377	八进制整数（以 0 开头）	255
0.0001	浮点数	0.0001
0xff	十六进制整数（以 0 和 x 开头）	255
0x3.45e2	错误。十六进制数不能有小数部分	N/A（编译错误）

3）布尔型

Boolean（布尔）数据类型只有两个值，它们是文字 true 和 false。Boolean 值是一个真值，它表示一个状态的有效性（说明该状态为真或假）。

脚本中的比较通常得到一个 Boolean 结果。考虑下一行 JavaScript 代码：

```
y = (x == 2000);
```

这里要比较变量 x 的值是否与数字 2000 相等。如果相等，比较的结果为 Boolean 值 true，并将其赋给变量 y。如果 x 与 2000 不等，则比较的结果为 Boolean 值 false。

另外，在 JavaScript 中数据类型 null 只有一个值：null，其含义是“无值”或“无对象”。当某个变量返回 undefined 值表示对象属性不存在或声明了变量但从未赋值。

3．常量和变量

1）常量

JavaScript 中的常量以直接量的形式出现，即在程序中直接引用，如“欢迎”、26 等。常量值可以为整型、实型、逻辑型及字符串型。

2）变量

变量声明。使用变量之前先进行声明。可以使用 var 关键字来进行变量声明。

```
var count;                          //单个变量声明
var count, amount, level;           //多个变量声明
var count = 0, amount = 100;        //变量声明和初始化
```

如果在 var 语句中没有初始化变量，变量自动取 JavaScript 值 undefined。

变量命名。JavaScript 是一种区分大小写的语言。因此变量名称 myCounter 和变量名称 mYCounter 是不一样的。变量的名称可以是任意长度。创建合法的变量名称应遵循如下规则：

第一个字符必须是一个 ASCII 码（大小写均可），或一个下划线(_)。注意第一个字符不能是数字；后续的字符必须是字母、数字或下划线；变量名称一定不能是保留字。

【例 4.4】 合法变量名称示例。

```
_pagecount
Part9
Number_Items
```

【例 4.5】 无效变量名称示例。

```
99Balloons          //不能以数字开头
Smith&Wesson        //“&”符号不能用于变量名
```

当要声明一个变量并进行初始化，但又不想指定任何特殊值，可以赋值为 JavaScript 值 null。

【例 4.6】 null 赋值示例。

```
<script language="javascript">
var bestAge = null;
var muchTooOld = 3 * bestAge;
//alert 实现了在浏览器中弹出消息对话框的功能
alert(bestAge); //消息框显示 bestAge 为 null
```

```
alert(muchTooOld);//消息框显示 muchTooOld 的值为 0
</script>
```

如果声明了一个变量但没有对其赋值，该变量存在，其值为 JavaScript 值 undefined。

【例 4.7】 undefined 赋值示例。

```
<script language="javascript">
var currentCount;
var finalCount = 1 * currentCount;
alert(currentCount);   //消息框显示 currentCount 为 undefined
alert(finalCount);     //消息框显示 finalCount 的值为 NaN （Not a Number）
</script>
```

对比例 4.6 和例 4.7 可以看出，JavaScript 中 null 和 undefined 的主要区别是 null 的操作如同数字 0，而 undefined 的操作如同特殊值 NaN（不是一个数字）。对 null 值和 undefined 值作比较总是相等的。

JavaScript 支持隐式声明，即可以不用 var 关键字声明变量，例如：

```
noStringAtAll = ""; //隐式声明变量 noStringAtAll
```

不能使用未经过声明的变量。例如：

```
var volume = length * width; //错误! length 和 width 不存在
```

4．运算符和表达式

1）运算符

JavaScript 运算符包括算术、逻辑、位、赋值以及其他运算符。运算符描述如表 4-2 所示。

表 4-2　运算符描述

算术运算符		逻辑运算符		位运算符		赋值运算符		其他运算符	
描　述	符号	描　述	符号	描　述	符号	描　述	符号	描　述	符号
负值	–	逻辑非	!	按位取反	~	赋值	=	删除	delete
递增	++	小于	<	按位左移	<<	运算赋值	op=	typeof	typeof
递减	– –	大于	>	按位右移	>>			void	void
乘法	*	小于等于	<=	无符号右移	>>>			instance of	instance of
除法	/	大于等于	>=	按位与	&			new	new
取模运算	%	等于（恒等）	==	按位异或	^			in	in
加法	+	不等于	!=	按位或	|				
减法	–	逻辑与	&&						
		逻辑或	||						
		条件运算符	?:						
		逗号	,						
		严格相等	===						
		非严格相等	!==						

相等（恒等）“==”与严格相等“===”的区别在于恒等运算符在比较前强制转换不同类型的值。例如，恒等对字符串"1"与数值 1 的比较结果将为 true。而严格相等不强制转换不同类型的值，因此它认为字符串"1"与数值 1 不相同。

字符串、数值和布尔值是按值比较的。如果它们的值相同，比较结果为相等。对象（包括 Array、Function、String、Number、Boolean、Error、Date 以及 RegExp 对象）按引用比较。即使这些类型的两个变量具有相同的值，只有在它们正好为同一对象时比较结果才为 true。

【例 4.8】 比较运算符示例。

```
<script language="javascript">
//具有相同值的两个基本字符串。
var string1 = "Hello", string2 = "Hello";
//具有相同值的两个 String 对象。
var StringObject1 = new String(string1),  StringObject2 = new String(string2);
var myBool=(string1 == string2);
alert(myBool); //消息框显示比较结果为 true
var myBool=(StringObject1 == StringObject2);
alert(myBool); //消息框显示比较结果为 false
//要比较 String 对象的值，用 toString() 或者 valueOf()方法
var myBool=(StringObject1.valueOf() == StringObject2);
alert(myBool); //消息框显示比较结果为 true
</script>
```

2）表达式

JavaScript 的表达式由常量、变量、运算符和表达式组成，有以下 3 类表达式。

- 算术表达式。值为一个数值型值，例如：5+a–x。
- 字符串表达式。值为一个字符串，例如："字符串 1"+str。
- 布尔表达式。值为一个布尔值，例如：(x= =y)&&(y>=5)。

JavaScript 解释器具有数据类型强制转换功能。对于强类型语言，如 C++、C#等，如果表达式不经过强制转换就试图对两个不同的数据类型（如一个为数字，另一个为字符串）执行运算，将产生错误结果。但在 JavaScript 中情况就不同了。

JavaScript 是一种自由类型的语言。它的变量没有预定义类型。JavaScript 变量的类型取决于它所包含值的类型，即赋给变量的值为小数，则变量为浮点型。在 JavaScript 中，可以对不同类型的值执行运算，不必担心 JavaScript 解释器产生异常。JavaScript 解释器自动将数据类型强制转换为另一种数据类型，然后执行运算。数据类型转换过程如表 4-3 所示。

表 4-3　数据类型转换

运　　算	结　　果	例　　子
数值与字符串相加	将数值强制转换为字符串	55 + "45" ➔ "5545"
布尔值与字符串相加	将布尔值强制转换为字符串	true + "45"➔　" true45"
数值与布尔值相加	将布尔值强制转换为数值。true=1；false=0	55 * true ➔ 55

要想显式地将字符串转换为整数，使用parseInt 方法。要想显式地将字符串转换为数

字，使用 parseFloat 方法。

【例 4.9】 显式地将字符串转换为数值。

55 + parseInt("45") ➔ 100　　**55 + parseInt("45AB") ➔ 100**

55 + parseInt("A45B") ➔ NaN　　**55 + parseInt("0xFF") ➔ 310**

55 + parseFloat("45.05") ➔ 100.05　　**(55=='55') ➔ true**

5．函数

函数为程序设计人员提供了实现模块化的工具。通常根据所要完成的功能，将程序划分为一些相对独立的部分，每一部分编写一个函数，从而使各部分充分独立，任务单一，结构清晰。JavaScript 函数定义语法格式为：

function 函数名(形式参数表){

//函数体

}

函数调用语法格式如下：

函数名(实参表);

当函数没有返回值时，可以不使用 return 语句，若使用 return，也只能使用不带参数的形式；当函数有返回值时，使用 return 语句返回函数值，格式为：

return 表达式 或 **return (表达式)**

JavaScript 支持两种函数：语言内部函数和自定义函数。

JavaScript 语言包含很多内部函数，例如数学函数 sin、cos、tan、sqrt、floor 等、字符串处理函数如 indexOf、lastIndexOf、length、replace、split、substring、toUpperCase、toLowerCase 等。某些函数可以操作表达式和特殊字符，例如非常有用的内部函数是 eval()。该函数可以对以字符串形式表示的任意有效的 JavaScript 代码求值。eval() 函数有一个参数，该参数就是想要求值的代码。

【例 4.10】 一个使用内部函数 eval()的示例。

```
<script language="javascript">
var iNumber="100";
var anExpression = "(16 * 9 % 7)";
var total = eval(anExpression + "/" + iNumber); //等同于求(16*9%7)/100 的值
alert(total);//将变量 total 赋值为 0.04。
</script>
```

在必要的时候，可以创建自定义函数。一个函数的定义中包含了一个函数语句和一个 JavaScript 语句块。

【例 4.11】 设计一个显示指定数的阶乘值的程序。

```
<html><head><title>函数示例</title></head>
<script language="JavaScript">
 function factor(num)
 {  var i,fact=1;
    for (i=1;i<num+1;i++) fact=i*fact;
    return fact;
 }
```

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

```
<script language="JavaScript">
//调用 factor 函数
alert("5 的阶乘="+ factor(5));  //显示“的阶乘=120”
</script>
</body></html>
```

使用函数要注意：(1)函数定义位置。语法上允许在 HTML 文件的任意位置定义和调用函数，建议在文件头部定义所有的函数，可以保证函数定义先于其调用语句载入浏览器，避免出现调用函数时由于函数定义尚未载入浏览器而引起的未定义错误。(2)函数参数。在定义函数时，可以给出一个或多个形式参数；调用函数时，却不一定要给出同样多个参数。在 JavaScript 中，系统变量 arguments.length 中保存了调用者给出的实参个数。(3)变量的作用域。在函数内用 var 保留声名的变量是局部变量，作用域仅局限于该函数；在函数外用 var 声明的变量是全局变量，作用域是整个 HTML 文件。函数内未用 var 声明的变量也是全局变量。局部变量与全局变量同名时，其操作互不影响。

【例 4.12】 默认求 1+2+…+1000，否则按指定开始值、结束值求和。

```
<html><head>
<script language="javascript">
function sum(StartVal,EndVal){
var ArgNum=sum.arguments.length;
var i,s=0;
if (ArgNum==0) { StartVal=1;EndVal=1000;}
else if (ArgNum==1) EndVal=1000;
for (i=StartVal;i<=EndVal;i++) s+=i;
return s;
}
</script>
</head>
<body>
<script language="javascript">
//document.write 表示在浏览器中输出文本
document.write("不给出参数调用函数 sum:",sum(),"<br>");
document.write("给出一个参数调用函数 sum:",sum(500),"<br>");
document.write("给出两个参数调用函数 sum:",sum(1,50),"<br>");
</script>
</body></html>
```

在浏览器中显示结果如图 4-1 所示。

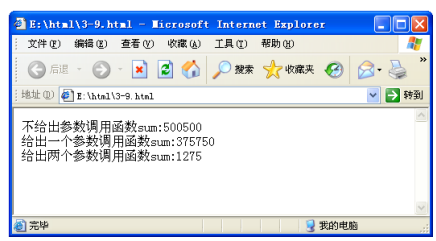

图 4-1　创建 JavaScript 函数

6．流程控制

1）if 条件语句

语法格式为：

if（条件）{
执行语句 1
}
else{
执行语句 2
}

其中条件可以是逻辑或关系表达式，若为 true，则执行语句 1；否则执行语句 2。若 if 后的语句有多行，则必须使用花括号将其括起来。

if 语句可以嵌套，格式为：

if（条件 1）执行语句 1；
else if（条件 2）执行语句 2；
else if（条件 3）执行语句 3；
⋮
else 执行语句；

在这种情况下，每一级的条件表述式都会被计算，若为真，则执行其相应的语句，否则执行 else 后的语句。

2）for 循环语句

语法格式为：

for（初始设置；循环条件；更新部分）{
语句集
}

初始设置告诉循环的开始位置，必须赋予变量的初值；循环条件用于判别循环停止时的条件。若条件满足，则执行循环体，否则跳出。更新部分定义循环控制变量在每次循环时按什么方式变化。初始设置、循环条件、更新部分之间，必须使用分号分隔。

for 循环的另一种用法是针对某对象集合中的每个对象或某数组中的每个元素，执行一个或多个语句。

for (变量 in 对象或数组) {
语句集
}

3）while 循环语句

语法格式为：

while（条件）{
语句集
}

当条件为真时，反复执行循环体语句，否则跳出循环体。循环体中必须设置改变循环条件的操作，使之离循环终止更近一步。

for 与 while 两种语句都是循环语句，使用 for 语句在处理有关数字时更易看懂，也较紧凑；而 while 循环更适合复杂的语句。

4）break 和 continue 语句

与 C++语言相同，使用 break 语句使得循环从 for 或 while 中强制跳出，continue 使得跳过循环内剩余的语句，并没有跳出循环体。

5）try...catch...finally 语句

该语句实现 JavaScript 错误处理。语法为：

```
try {
    测试语句集}
catch(exception){
    错误处理语句集}
finally {
   无错误处理语句集}
```

7．事件驱动及事件处理

事件（events）是指对计算机进行一定的操作而得到的结果，例如将鼠标移到某个超链接上、单击鼠标按钮等都是事件。由鼠标或热键引发的一连串程序的动作，称为事件驱动（Event Driver）。对事件进行处理的程序或函数，称为事件处理程序（Event Handler）。

在 HTML 文件中，可用支持事件驱动的 JavaScript 语言编写事件处理程序。用 JavaScript 进行事件编程主要用于两个目的：

- 验证用户输入窗体的数据；
- 增加页面的动感效果。

一个 HTML 元素响应鼠标的事件和键盘的事件如表 4-4 所示。某些鼠标事件虽事件名称不一样，但响应效果几乎一样，用户可根据实际需要选择某个事件进行编程。

表 4-4　鼠标事件和键盘事件列表

事件名称	说　明	事件名称	说　明	事件名称	说　明
onclick	鼠标左键单击	ondblclick	鼠标左键双击	onmouseup	松开鼠标左键或右键
onmousedown	按下鼠标左键或右键	onmouseover	鼠标指针在该 HTML 元素经过	onmouseout	鼠标指针离开该 HTML 元素
onmousemove	鼠标指针在其上移动时	onmousewheel	滚动鼠标滚轮	onfocus	当用鼠标或键盘使该 HTML 元素得到焦点时
onkeypress	击键操作发生时	onkeyup	松开某个键时	onkeydown	按下某个键时
onchange	当文本框的内容发生改变的时候	onselect	当用鼠标或键盘选中文本时	onblur	HTML 元素失去焦点时

【例 4.13】 鼠标单击事件。

```
<html><body>
<form>
<Input type="button" Value="鼠标响应" onclick="alert(这是一个例子')">
</form>
</body></html>
```

当鼠标单击“鼠标响应”按钮的时候，自动弹出一个 alert 对话框。试着将 onclick 事件名称换成其他事件名称，查看对应事件响应结果。从该例可知，事件编程的语法格式为：

事件名称 = 函数名 或 处理语句 或 函数

例 4.14、例 4.15 和例 4.16 分别是 JavaScript 事件编程的三种方法，它们的执行效果完全相同，用户在文本框中输入数字或字符后，单击“检查”按钮检查输入的字符串是否全由数字组成，显示 true 或 false。注意我们为文本框设置属性 ID 为 mytext，要取到文本框用户输入的内容，用 mytext.value 即可。

例 4.15 的事件编程用于执行代码较少的情况。当执行代码较多情况下采用例 4.14 的事件编程方式，可读性好，便于程序维护和代码重用。注意在例 4.16 中，鼠标单击不能用 onclick，只能用 onmousedown 或 onmouseup，且 JavaScript 脚本程序必须放在按钮和文本框定义之后，否则就会出现没有事先定义而在脚本程序中引用的错误。例 4.14 中的脚本程序放在按钮和文本框定义前或后均可。

【例 4.14】 鼠标单击（函数名）。

```
<html>
<head><title>检查输入的字符串是否全由数字组成</title></head>
<script language="javascript">
  function checkNum(str)
  { var TestResult = !/\D/.test(str); //使用正则表达式测试字符串是否全由数字组成
    alert(TestResult);
  }
</script>
<body>
<input id="mytext" type="text" value='12332'>
<input id="mybut" type="button" value="检查"  onclick="checkNum(mytext.value)">
</body></html>
```

【例 4.15】 鼠标单击（处理语句）。

```
<html>
<head><title>检查输入的字符串是否全由数字组成</title></head>
<body>
<input id="mytext" type="text" value='12332'>
<input  id="mybut"  type="button"  value=" 检 查 "  onclick="javascript:var
TestResult=!/\D/.test(mytext.value); /*使用正则表达式测试字符串*/ alert
(TestResult);">
</body></html>
```

【例 4.16】 鼠标单击（函数）。

```
<html>
<head><title>检查输入的字符串是否全由数字组成</title></head>
<body>
```

```
<input id="mytext" type="text" value='12332'>
<input id="mybut" type="button" value="检查">
```

```
<script language="javascript">
mybut.onmousedown=function() { /*mybut 为按钮的 ID*/
  var TestResult=!/\D/.test(mytext.value); /*使用正则表达式测试字符串是否全是数字*/
   alert(TestResult);
}
</script>
</body></html>
```

【例 4.17】 onchange、onselect、onfocus 事件例子。

onchange 事件是当某个 HTML 元素（例如文本框）的内容改变的时候发生的事件；onselect 事件就是当某个 HTML 元素（例如文本框）的文本内容被选中的时候发生的事件；onfocus 事件就是当光标落在某个 HTML 元素，使它得到焦点的时候发生的事件。

此例中，当在文本框 Text1 中输入内容或删除某个字符后，一旦失去焦点就会自动弹出一个“文本发生变化!”alert 框。当在文本框 Text2 中用键盘或鼠标选中文本的时候就会自动弹出“我被选中!”alert 框。当用鼠标选中文本框 Text3 的时候，触发 onfocus 事件，弹出“Test3 得到焦点!”alert 框。

```
<html><body>
<form>
<input id="Test1" type="text" value="Test" onChange='alert("文本发生变化!")'>
<input id="Test2" type="text" value="Test" onSelect='alert("我被选中!")'>
<input id="Test3" type="text" value="Test1" onFocus='alert("Test3 得到焦
点!")'>
<br>
<input id="Test4" type="text" value="Test2" onFocus='alert("Test4 得到焦
点!")'>
</form>
</body></html>
```

4.1.3　JavaScript 对象编程技术

1．JavaScript 的对象

JavaScript 并不完全支持面向对象的程序设计方法，例如它没有提供抽象、继承、封装等面向对象的基本属性。但它支持开发对象类型及根据对象产生一定数量的实例，同时还支持开发对象的可重用性，实现一次开发、多次使用的目的。

JavaScript 中的对象是由属性（properties）和方法（methods）两个基本的元素构成的。在 JavaScript 中使用一个对象可采用以下三种方式获得：

- 引用 JavaScript 内置对象，如 Date、Math、String 等；
- 用户自定义对象；
- 引用浏览器对象，下一节将专门介绍。

2．JavaScript 常用的内置对象

1）Array 对象

可用 Array 对象创建数组。数组是若干元素的集合，每个数组都用一个名字作为标识。JavaScript 中没有提供明显的数组类型，可通过 JavaScript 内建对象 Array 和使用自定义对象的方式创建数组对象。

【例 4.18】 使用 JavaScript 内建对象 Array 生成一个新的数组。通过 new 保留字来创建数组对象，其语法格式为：var 数组名=new Array（数组长度值）。

```
var theMonths = new Array(6); //创建数组对象 theMonths，具有 6 个数组元素
theMonths[0] = "Jan";
theMonths[1] = "Feb";
theMonths[2] = "Mar";
theMonths[3] = "Apr";
theMonths[4] = "May";
theMonths[5] = "Jun";
```

下面的示例与上一个示例是等价的。

```
var theMonths = new Array("Jan", "Feb", "Mar", "Apr", "May", "Jun");
```

数组创建后，可通过[ ]来访问数组元素，用数组对象的属性 length 可获取数组元素的个数。当向用关键字 Array 生成的数组中添加元素时，JavaScript 自动改变属性 length 的值。JavaScript 中的数组索引总是以 0 开始，而不是 1。例如：

```
var my_array = new Array();    //创建动态数组
for (i = 0; i < 10; i++)
   {
   my_array[i] = i;
   }
x = my_array[4];
alert("x="+x);                 //输出 4
alert(my_array.length);        //输出数组的元素个数为 10
```

【例 4.19】 使用自定义对象的方式创建数组对象。通过 function 定义一个数组，并使用 new 对象操作符创建一个具有指定长度的数组。其中 arrayName 是数组名，size 是数组长度，通过 this[i]为数组赋值。定义对象后还不能马上使用，还必须使用 new 操作符创建一个数组实例 MyArray。一旦给数组赋予了初值后，数组中就具有真正意义的数据了，以后就可以在程序设计过程中直接引用。

```
<script language="javascript">
function arrayName(size){
this.length=size;
for(var i=0; i<=size;i++) this[i]=0;
return this;
```

```
}
var  MyArray = new arrayName(10);
MyArray[0]=1; MyArray[1]=2;MyArray[2]=3;
```

```
MyArray[3]=4;MyArray[4]=5;MyArray[5]=6;
MyArray[6]=7;MyArray[7]=8;MyArray[8]=9;
MyArray[9]=10;
alert(MyArray[7]); //输出 8
</script>
```

2）String 对象

在 JavaScript 中，可以将字符串当作对象来处理。创建 String 对象实例。格式为：

[var] String 对象实例名 = 字符串值；

String 对象只有一个属性，即 length 属性，包含了字符串中的字符数（空字符串为 0），它是一个数值，可以直接在计算中使用。

String 对象内置方法有三十多种。例如 anchor、link、substring、indexOf、replace 等。

【例 4.20】 String 对象的建立和使用。

```
<script language="javascript">
//设置变量 howLong 为 11
var howLong = "Hello World".length;
//锚点方法 anchor()。使用 anchor 作用与 Html 中(A Name="")一样
//格式为: string.anchor(anchorName)
//创建一个名为 start 的锚点，该处显示文字"开始"
var astr="开始";
var aname=astr.anchor("start");
document.write(aname);

//超链接方法 link()。用于创建一个超链接，与 HTML 中(A href="")作用相同
//格式为:string.link(URL)
var hstr="重庆大学";
var hname=hstr.link("http://www.cqu.edu.cn");
document.write(hname);

//substring()方法:substring(start,end)。它返回字符串的一部分,该字符串包含从 start
//直到 end（不包含 end）的子字符串
//substring 方法使用 start 和 end 两者中的较小值作为子字符串的起始点。例如，
//strvar.substring(0, 3) 和 strvar.substring(3, 0) 将返回相同的子字符串
//如果 start 或 end 为 NaN 或负数，那么它将被替换为 0
//子字符串的长度等于 start 和 end 之差的绝对值。例如，在 strvar.substring(0, 3) 和
//strvar.substring(3, 0) 中，返回的子字符串的长度为 3

//字符搜索: indexOf[str,fromIndex]
Str1="0123456789";//创建一个 string 对象 Str1
Str2="2345";//创建一个 string 对象 Str2
var aChunk = Str1.substring(4, 7); //将 aChunk 设为"456"
```

```
document.write("aChunk="+aChunk);
found= Str1.indexOf(Str2);//返回 Str2 在 Str1 中的起始位置

//创建字符串对象的另外一种方法是用 new
```

```
var mystr= new  String("<br>重庆大学<br>");
document.write(mystr.link("http://www.cqu.edu.cn"));
</script>
```

3）Math 对象

Math 对象提供了常用的数学函数和运算，如三角函数、对数函数、指数函数等。Math 中提供了 6 个属性，它们是：常数 E；以 10 为底的自然对数 LN10；以 2 为底的自然对数 LN2；圆周率 PI：近似值为 3.142；1/2 的平方根 SQRT1-2；2 的平方根为 SQRT2；以 10 为底，E 为对数的 LOG10E；以 2 为底，E 为对数的 LOG2E。主要方法有绝对值 abs()、正弦 sin()、余弦 cos()、反正弦 asin()、反余弦 acos()、正切 tan()、反正切 atan()、四舍五入 round()、平方根 sqrt()、乘幂值 pow(base,exponent)；返回 0 与 1 之间的随机数 random()等。

【例 4.21】 Math 对象的使用。

```
<script language="javascript">
var radius = 5;                                    // 声明一个半径变量并赋数值
var circleArea = Math.PI * radius * radius;        // 注意 Math 和 PI 需大写
// 本公式计算给定半径的球体的体积。
volume = (4/3)*(Math.PI*Math.pow(radius,3));
alert(volume);                                     //输出.5987

//也可用 With 保留字来简化程序的写法
with (Math){
var circleArea = PI * radius * radius;     // 注意 Math 和 PI 需大写
// 本公式计算给定半径的球体的体积。
volume = (4/3)*(PI* pow(radius,3));
}
alert(volume);                                     //输出.5987
</script>
```

4）Date 对象

Date 对象可以被用来表示任意的日期和时间，获取当前系统日期以及计算两个日期的间隔等。常用的方法有 getFullYear、getMonth、getDate 等。通常 Date 对象给出星期、月份、天数和年份以及以小时、分钟和秒表示的时间。该信息是基于 1970 年 1 月 1 日 00:00:00.000 GMT 开始的毫秒数，其中 GMT 是格林威治标准时间（首选术语是 UTC，Universal Coordinated Time，或者“全球标准时间”，它引用的信号是由“世界时间标准”发布的）。JavaScript 可以处理 250000 B.C.到 255000 A.D 范围内的日期。可使用 new 运算符创建一个新的 Date 对象。

【例 4.22】 Date 对象的使用。

```
<script language="javascript">
/*
本示例使用前面定义的月份名称数组。
第一条语句以“Day Month Date 00:00:00 Year”格式对 Today 变量赋值。 */
```

```
var Today = new Date();  //获取今天的日期
// 提取年，月，日
thisYear = Today.getFullYear(); thisMonth = Today.getMonth(); thisDay =
Today.getDate();
// 提取时，分，秒
thisHour=Today.getHours(); thisMinutes=Today.getMinutes();thisSeconds=Today.
getSeconds();

//提取星期几
thisWeek=Today.getDay();
var x = new Array("日", "一", "二"); x = x.concat("三","四", "五", "六");
thisWeek=x[thisWeek];

nowDateTime="现在是"+thisYear+"年"+thisMonth +"月"+thisDay+"日";
nowDateTime+=thisHour+"时"+thisMinutes+"分"+thisSeconds+"秒";
nowDateTime+="星期"+thisWeek;

document.write(nowDateTime+"<br>");   //输出：现在是年月日时分秒

//计算两个日期相差的天数
var datestring1 = "November 1, 1997 10:15 AM";
var datestring2 = "December 1, 2007 10:15 AM";
var DayMilliseconds= 24*60*60*1000;    //1 天的毫秒数
var t1 = Date.parse(datestring1);      //换算成自年月日到年月日的毫秒数
var t2 = Date.parse(datestring2);      //换算成自年月日到年月日的毫秒数
   s = "There are "
   s += Math.round(Math.abs((t2-t1)/DayMilliseconds)) + " days "
   s += "between " + datestring1 + " and " + datestring2 ;

document.write(s); //输出: There are 3682 days between November 1, 1997 10:15
                   //AM and December 1, 2007 10:15 AM
</script>
```

5）Number 对象

除了 Math 对象中可用的特殊数值属性（例如 PI）外，Number 对象有几个其他的数值属性，如表 4-5 所示。

表 4-5　Number 对象数值属性

属　性	描　述	属　性	描　述
MAX_VALUE	数值最大数	POSITIVE_INFINITY	代表正无穷大
MIN_VALUE	数值最小数	NEGATIVE_INFINITY	代表负无穷

NaN	特殊非数量值		

Number.NaN 是一个特殊的属性，被定义为“不是数值”。例如被 0 除返回 NaN。试图解析一个无法被解析为数字的字符串同样返回 Number.NaN。把 NaN 与任何数值或本身作比较的结果都是不相等。不能通过与 Number.NaN 比较来测试 NaN 结果，而应该使用 isNaN() 函数。

6）JavaScript 中的预定义函数

它提供了与任何对象无关的系统函数，使用这些函数不需创建任何实例，可直接用。例如：

返回字符串表达式中的值。eval（字符串表达式），例：test=eval("8+9+5/2")

返回字符的编码。 escape(string) //用%xx 十六进制形式编码

返回字符串 ASCII 码。 unescape(string) //将用 escape 编码过的字串复原

返回实数。 parseFloat(floatstring) //字符数字变成实数

返回不同进制的数。 parseInt(numbestring,radix) //radix 是数的进制，
//numbestring 字符串数。字符数字按进制变成整数。

3．用户自定义对象

使用 JavaScript 可以创建自己的对象。虽然 JavaScript 内部和浏览器本身的功能已十分强大，但 JavaScript 本身还提供了创建新对象的方法，可以完成许多复杂的工作。

要创建一个对象，必须首先为其定义一个构造函数，赋予对象属性和方法。其基本格式如下：

```
function ObjectName（属性表）{
this.property1=property1;          //属性的定义
this.property2=property2;          //属性的定义
⋮
this.method1=FunctionName1;        //方法的定义
this.method2=FunctionName2;        //方法的定义
⋮
}
```

【例 4.23】 创建对象 pasta。

```
<script language="javascript">
//定义 pasta 对象
function pasta(grain, width, shape, hasEgg) {
  this.grain = grain;
  this.width = width;
  this.shape = shape;
this.hasEgg = hasEgg;
}
//使用对象的时候，用 new 进行实例化，下面建立了两个对象实例。
var spaghetti = new pasta("wheat", 0.2, "circle", true);
var linguine = new pasta("wheat", 0.3, "oval", true);
```

```
document.write(spaghetti.shape)        //输出 circle
document.write(linguine.shape)         //输出 oval
</script>
```

【例 4.24】 扩充上例中定义的 pasta 构造函数以包含 toString 方法。

```
<script language="javascript">
//定义 pasta 对象
function pasta(grain, width, shape, hasEgg) {
   this.grain = grain;
   this.width = width;
   this.shape = shape;
   this.hasEgg = hasEgg;
   this.toString = pastaToString;
}
function pastaToString(){
    return "Grain: " + this.grain + "\n" +"Width: " + this.width + "\n" +
       "Shape: " + this.shape + "\n" +"Egg?: " + Boolean(this.hasEgg);
}
//用 new 建立 pasta 对象的实例 spaghetti
var spaghetti = new pasta("wheat", 0.2, "circle", true);

//可以给对象实例 spaghetti 添加属性,以改变该实例
spaghetti.color = "pale straw";
spaghetti.drycook = 7;
spaghetti.freshcook = 0.5;

alert(spaghetti); //输出: Grain: wheat Width: 0.2 Shape: circle Egg?: true
alert(spaghetti.freshcook)//输出: .5

//用 new 建立 pasta 对象的实例 chowFun
var chowFun = new pasta("rice", 3, "flat", false);

//chowFun 实例中并不包括实例 spaghetti 添加的 color 属性。可以为 pasta 对象添加一个
//color 属性，以后所有 pasta 的对象实例均可使用此属性
pasta.prototype.color = "yellow";

alert(spaghetti.color);    //输出: pale straw
alert(chowFun.color);      //输出: yellow （chowFun 已经具有 color 属性了）
</script>
```

4.1.4 JavaScript ActiveX 编程技术

在第 3 章已经介绍了在 HTML 页面中如何使用 ActiveX 控件播放声音和 Flash 等。如何在 JavaScript 脚本程序中进行 ActiveX 控件的编程处理呢？一般来说，在计算机上安装好系统软件和应用软件后，一些 ActiveX 控件就会安装在计算机上，就可以直接利用这些 ActiveX 控件来实现所需要的功能。例如 FileSystemObject 控件对象提供对计算机文件系统的访问；Excel.Application 和 Word.Application 分别提供对 Excel 和 Word 的控制和操作。JavaScript 提供了 ActiveXObject 方法实现对 ActiveX 控件的访问。

FileSystemObject 控件对象提供了几乎所有访问磁盘文件系统所需要的功能，例如文件与文件夹的创建和删除、复制文件、删除文件、移动文件、驱动器操作、读写文件操作等等，具体的属性、方法和事件用法可参阅相关资料。

下面主要通过例子的方式说明 JavaScript 的 ActiveX 编程技术。

【例 4.25】 文件系统操作。

```
<script type="text/javascript" >
var fso = new ActiveXObject("Scripting.FileSystemObject")
fso.CreateFolder ("C:\\Bonus");        //在 C 盘创建一个文件夹
fso.DeleteFolder ("C:\\Bonus");        // 删除创建的文件夹
fso.CopyFile("c:\\temp\\11.bmp","c:\\Bonus\\22.bmp"); //复制文件
// 创建新文件
var tf = fso.CreateTextFile("c:\\testfile.txt", true);
tf.WriteLine("Testing 1, 2, 3.") ;     // 填写数据，并增加换行符
tf.WriteBlankLines（3）;               // 增加 3 个空行
tf.Write ("This is a test.");          // 填写一行，不带换行符
tf.Close();                            // 关闭文件
// 打开文件
var ForReading=1;
var ts = fso.OpenTextFile("c:\\testfile.txt", ForReading);
s = ts.ReadLine();                     // 读取文件一行内容到字符串
alert("File contents = '" + s + "'"); // 显示字符串信息
ts.Close();                            // 关闭文件
</script>
```

【例 4.26】 调用 Excel 的例子 1。

```
<script language="javascript">
//启动创建对象的应用程序 Excel
var ExcelApp=new ActiveXObject("Excel.Application");
var ExcelSheet = new ActiveXObject("Excel.Sheet");    //创建 Excel 工作表
ExcelSheet.Application.Visible = true;                //使 Excel 窗口可见
// 将一些文本放置到表格的第一格中
ExcelSheet.ActiveSheet.Cells(1,1).Value = "This is column A, row 1";
```

```
// 保存 EXCEL 到 C:\TEST.XLS
ExcelSheet.SaveAs("C:\\TEST1.XLS");
//用 Quit 方法关闭 Excel
ExcelApp.Quit();
//可以找到 C:\TEST1.XLS 文件，检查其正确性
</script>
```

【例 4.27】 调用 Excel 的例子 2。

```
<script language="javascript">
//启动 Excel
var ExcelApp = new ActiveXObject("Excel.Application");
ExcelApp.Visible = true;   //使 Excel 窗口可见
ExcelApp.WorkBooks.Open("c:\\TEST.xls");
var objExcelBook=ExcelApp.ActiveWorkBook;
var objExcelSheets=objExcelBook.Worksheets;
var objExcelSheet=objExcelBook.Sheets(1); //指定当前工作区为 Sheet2
//此处为对 Excel 单元格进行填写数据的语句
objExcelSheet.Range("B2:k2").Value=Array("Week1","Week2","Week3","Week4",
"Week5","Week6","Week7");
objExcelSheet.Range("B3:k3").Value=Array("67","87","5","9","7","45","45",
"54","54","10");
objExcelSheet.Range("B4:k4").Value=Array("10","10","8","27","33","37","50",
"54","10","10");
objExcelSheet.Range("B5:k5").Value=Array("23","3","86","64","60","18","5",
"1","36","80");
objExcelSheet.Cells(3,1).Value="InternetExplorer";
objExcelSheet.Cells(4,1).Value="Netscape";
objExcelSheet.Cells(5,1).Value="Other";
objExcelSheet.SaveAs("C:\\TEST2.XLS");    //保存到 C:\TEST2.XLS
ExcelApp.Quit();                          //退出 Excel
</script>
```

【例 4.28】 调用 Word 的例子。

```
<script language="javascript">
WordApp = new ActiveXObject("Word.Application");  //启动 Word
  WordApp.Application.Visible = true;             //使 Word 窗口可见
  var mydoc=WordApp.Documents.Add("",0,1);        //新建一个文档
WordApp.ActiveWindow.ActivePane.View.Type=3;      //Word 视图模式为页面

WordApp.Selection.TypeText("测试案例");           //输入字符串
  WordApp.Selection.HomeKey (5,1);                //光标移到行首
WordApp.Selection.Font.Bold = 9999998 ;           //wdToggle
WordApp.Selection.WholeStory();                   //选中整个文档内容
```

```
  mydoc.SaveAs("c:\\test.doc");                    //存盘到 c:\test.doc
for(i=WordApp.Documents.Count;i>0;i--){            //关闭所有打开的 Word 文档
  WordApp.Documents(i).Close(0);
 }
 WordApp.Application.quit();  //退出 Word
</script>
```

一般的参考书很少涉及详细讲解如何使用 Excel 和 Word 的方法和属性，使得开发者在开发 Word 和 Excel 的应用中望而生畏。其实，可以在 Word 或 Excel 中通过录制宏的功能先将你准备在 Word 或 Excel 中需要操作的过程通过录制宏的方式录制下来，然后打开录制的宏，它们是使用 Visual Basic Application 语言编写的程序，将这些程序复制到 JavaScript 中加以改造，即可完成在 JavaScript 中的编程。其中的参数可在 Word 或 Excel 中，通过单击“工具”|“宏”|“Visual Basic 编辑器”菜单打开 Visual Basic 编辑器，按下 F2 键打开对象浏览器，输入参数后，就可得到参数对应的值。例如字体的加粗与否通过参数 wdToggle 设定，查询出来的参数值为 9999998。

4.2 HTML DOM 程序设计初步

4.2.1 HTML 文档对象模型

HTML 文档对象模型 DOM（HTML Document Object Model）是一个能够让程序和脚本动态访问和更新 HTML 文档内容、结构和样式的语言平台。HTML DOM 是一个跨平台、可适应不同程序语言的文件对象模型，它采用直观一致的方式，将 HTML 或 XHTML 文件进行模型化处理，提供存取和更新文档内容、结构和样式的编程接口。使用 DOM 技术，不仅能够访问和更新页面的内容及结构，而且还能操纵文档的风格样式。可以将 HTML DOM 理解为网页的 API。它将网页中的各个 HTML 元素看做一个个对象，从而使网页中的元素可以被 JavaScript 等语言获取或者编辑。

因为 DOM 规范在不断发展，各种浏览器对 DOM 的支持情况会有所变化，在使用时应参照最新的 DOM 文档。我们经常看到在某个浏览器下工作正常的页面在另外一种浏览器下却显示不正常，这就是浏览器对 DOM 的支持不尽相同。对于某些专业的大型网站，开发人员采用了识别浏览器类型，针对浏览器的不同而进行相应代码的处理，以尽量使生成的网页在各种浏览器上工作正常。

微软 IE 浏览器的文档对象模型如图 4-2 所示。

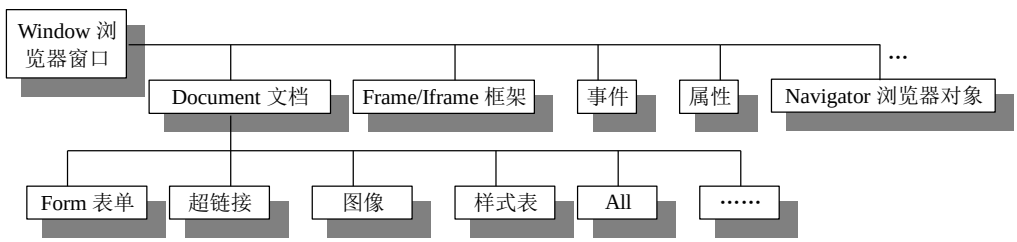

图 4-2　IE 浏览器的文档对象模型

HTML DOM 定义了一种访问并操作 HTML 文档的标准方法。HTML DOM 将 HTML 文档视为嵌套其他元素的树型结构元素。所有的元素，它们包含的文字以及它们的树型都可以被 DOM 树所访问到。它们的内容可以被修改和删除，并且可以通过 DOM 建立新的元素。

HTML DOM 所包含的内容实在太多，很难在短时间内掌握和学好 HTML DOM 编程，掌握 HTML DOM 编程最简单的方法就是多读别人的程序。当你访问某个网站的时候，如果你觉得网页上的某些功能不错，在 IE 浏览器中，可通过快捷菜单中的“查看源文件”或通过主菜单“查看”|“源文件”在文本编辑器中查看 HTML 页面文件，从中可以发现很多有用的代码。但很多客户端脚本程序通过<script src="myjsfile.js"></script>这样的方式引用，无法在文本编辑器中看到，实际上这些文件全部放在浏览器的缓存中（Web 服务器传回的各类文件，包括图片、页面文档等都放在缓存中）。可通过以下方法查看脚本程序源代码：

（1）在微软 IE 浏览器主菜单中，单击“工具”|“Internet 选项”，打开“Internet 选项”对话框，单击“删除文件”按钮后选择“删除所有脱机文件”单击“确定”按钮后退出对话框。这样做的目的就是可以让你方便地找到所需要的 js 文件。

（2）在浏览器中输入网址后，通过上述查看源文件方法，找到引用的 js 文件，不妨假设为 myjsfile.js。

（3）选择菜单“工具”|“Internet 选项”，打开“Internet 选项”对话框，单击“设置”按钮后出现“设置”对话框，单击“查看文件”按钮后，就会显示缓存文件夹 Temporary Internet Files，如图 4-3 所示。按下 F3 功能键或单击工具栏的“搜索”，输入在缓存文件夹中要搜索的文件名 myjsfile*.js（或输入*.js，注意文件名后需加上*），即可搜索出 myjsfile[1].js、myjsfile[2].js 等文件。浏览器缓存中存放了许多被访问网站的文件，很可能文件名有相同的，因此缓存中采取了在文件名后加上数字标识来区分不同网站的相同文件名。

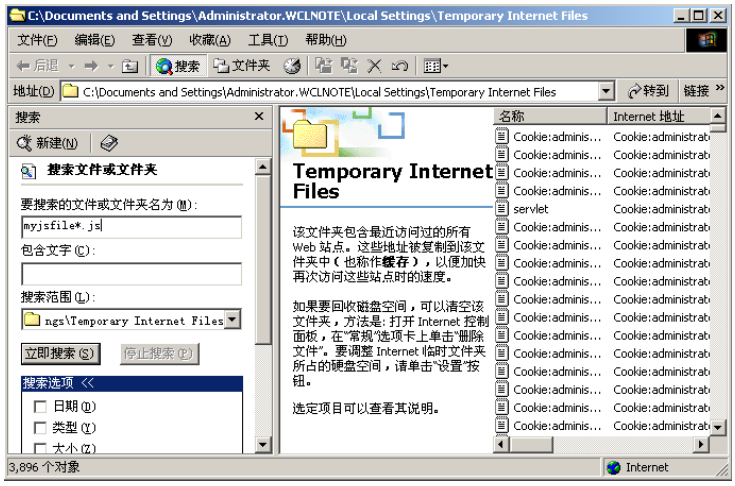

图 4-3　搜索 IE 浏览器缓存中的 js 文件

（4）打开此 js 文件就可看到相应的源代码。

利用此方法可以找到很多有用的文件，例如，对于一个 Flash 制作的 swf 文件，采用上述方法后可以将它从缓存中取出存放到另外一个文件夹作长期保存。

本节主要通过实例的方式让读者领悟 HTML DOM 编程过程。

4.2.2 通过 DOM 操纵 HTML 元素

可以这样来理解图 4-2 所示对象模型：在 HTML DOM 中，打开的浏览器窗口可看成 Window 对象，浏览器显示页面的区域可看成 document 对象，各种 HTML 元素就是 document 的子对象。

要对某个 HTML 元素进行操控，必须为它设置 ID 属性或 Name 属性。我们可以把某 HTML 元素的 ID 属性看成是该控件的名称，DOM 中通过 ID 属性或 Name 属性来操控 HTML 元素。建议全部用 ID 属性，而不用 Name 属性，Name 属性只是为了兼容低版本浏览器。例如：

指定 ID 属性：<input **id="myColor"** type="text" value="red" >

指定 Name 属性：<input **name="myColor"** type="text" value="red" >

例 4.29 中，用户在文本框 myColor 和 myTitle 中分别输入颜色和新的窗口标题，单击各自按钮就会看到浏览器中背景颜色和窗口标题发生的变化。注意，在 JavaScript 程序中，用 window.myColor.value 取 id 为 myColor 文本框的值，用 window.mybut1.onmousedown 表示 id 为 mybut1 按钮的单击事件。

在例 4.30 中，将例 4.29 中的 HTML 控件放在 id 为 myForm 的表单标记中，在 JavaScript 程序中，用 window.MyForm.myColor.value 取 id 为 myColor 文本框的值，用 window.MyForm.mybut1.onmousedown 表示 id 为 mybut1 按钮的单击事件。

这样就产生了一个问题，即在有表单和无表单标记对象的 HTML 文档中，对某个 HTML 元素对象的访问方式有很大变化，这对程序员来说会带来很大的麻烦。因此 HTML DOM 中提供了统一访问 HTML 元素的六种方法，它们的格式是：

- **window.document.all.item("HTML 元素的 ID")**

例如：window.document.all.item("myColor")。

- **document.all.HTML 元素的 ID**

例如：window.document.all.myColor。

- **window.document.getElementById("HTML 元素的 ID")**

例如：window.document.getElementById("myColor")。

- **window.document.getElementsByName("HTML 元素的 Name 属性值")**

例如：window.document.getElementsByName("firstName")。

- **window.document.all.namedItem("HTML 元素的 Id 或 Name 属性值")**

例如：window.document.all.namedItem ("myColor")。

- **window.document. getElementsByTagName("HTML 标记名称")**

例如：window.document.getElementsByTagName("div")。

getElementsByTagName 方法可实现当标记在没有定义 ID 或 Name 属性的情况下仍然可以被访问。例如：

```
<script language='javascript' >
       for(i=0;i<document.getElementsByTagName("div").length;i++) {
        var pp=document.getElementsByTagName("div")[i].innerText;
        alert(pp); } //显示所有文本块中的内容
</script>
```

注意：由于 HTML DOM 中默认根对象为 window，因此在 window 子对象和它的方法引用中可以省略 window。例如 window.document 可缩写成 document。

将例 4.29 和例 4.30 中的脚本程序分别换成例 4.31、例 4.32、例 4.33 中的脚本程序，可以看出，访问 HTML 元素的方法不同，而得到的结果相同。其中 getElementById 方法是 W3C 倡导的标准方法，各种浏览器均支持它。如果要设计兼容各种浏览器的网页，建议采用 getElementById 方法来访问 HTML 元素对象。

【例 4.29】 动态改变浏览器背景颜色和浏览器窗口标题 1。

```
<html>
<head><title>无标题页</title>
</head>
<body>
<input id="myColor" type="text" value="red" >
<input id="myTitle" type="text" value="新的窗口标题" >
<input id="mybut1" type="button" value="改变页面背景颜色">
<input id="mybut2" type="button" value="改变浏览器窗口标题">
<script language="javascript">
    window.mybut1.onmousedown=function(){window.document.bgColor=window.
myColor.value;}
    window.mybut2.onmousedown = function() {window.document.title=window.
myTitle.value;}
</script>
</body></html>
```

【例 4.30】 动态改变浏览器背景颜色和浏览器窗口标题 2。

```
<html>
<head><title>无标题页</title>
</head>
<body>
<form id="myForm" action="" method = "post" >
<input id="myColor" type="text" value="red" >
<input id="myTitle" type="text" value="新的窗口标题" >
<input id="mybut1" type="button" value="改变页面背景颜色">
<input id="mybut2" type="button" value="改变浏览器窗口标题">
</form>
<script language="javascript">
```

```
  window.myForm.mybut1.onmousedown= function() {
     window.document.bgColor = window.myForm.myColor.value;}
  window.myForm.mybut2.onmousedown= function(){
     window.document.title=window.myForm.myTitle.value;}
</script>
</body>
</html>
```

【例 4.31】 脚本程序 1——动态改变浏览器背景颜色和浏览器窗口标题。

```
<script language="javascript">
```

```
    var b1=window.document.all.item("mybut1");
    var b2=window.document.all.item("mybut2");
    var t1=window.document.all.item("myColor");
    var t2=window.document.all.item("myTitle");
    b1.onmousedown= function() {window.document.bgColor = t1.value;}
    b2.onmousedown= function() {window.document.title = t2.value;}
</script>
```

【例 4.32】 脚本程序 2——动态改变浏览器背景颜色和浏览器窗口标题。

```
<script language="javascript">
    var b1=window.document.all.mybut1;
    var b2=window.document.all.mybut2;
    var t1=window.document.all.myColor;
    var t2=window.document.all.myTitle;
    b1.onmousedown= function() { window.document.bgColor = t1.value;}
    b2.onmousedown= function() { window.document.title = t2.value;}
</script>
```

【例 4.33】 脚本程序 3——动态改变浏览器背景颜色和浏览器窗口标题。

```
<script language="javascript">
    var  b1=document.getElementById("mybut1");
   var b2=document.getElementById("mybut2");
    var t1=document.getElementById("myColor");
    var t2=document.getElementById("myTitle");
    b1.onmousedown= function() {document.bgColor = t1.value;}
    b2.onmousedown= function() {document.title = t2.value;}
</script>
```

4.2.3　HTML DOM 主要对象介绍及编程实例

1．窗口对象（window）

window 对象处于对象层次的最顶端，每个对象代表一个浏览器窗口，封装了窗口的方法和属性。window 对象所包含的属性、方法、事件、对象如图 4-4 所示。

编程人员可以利用 window 对象控制浏览器窗口显示的各个方面，如对话框、框架等。主要内容包括：

- onload 和 onunload 都是窗口对象事件，在加载 Web 页面到内存和从内存卸载 Web 页面时发生。
- close 方法可用来关闭浏览器窗口。open 方法可以打开一个新的浏览器窗口并加载指定的 Web 页，例如以下脚本实现了这样的功能：新打开一个浏览器窗口，窗口左上角相对屏幕左上角的坐标位置为（75，20）像素，窗口宽 480、高 420 个像素，在窗口加载 QQ 首页。其中 sName 可以取值为"replace"（用 qq 首页替换当前页面）、

图 4-4　window 对象

"_blank"（新打开一个浏览器窗口显示 QQ 首页）、"_parent"（在当前页的父窗口中打开 QQ 首页）、"_search"（在搜索窗口中打开 QQ 首页）、"_self"（用 QQ 首页替换当前页面）、"_top"（用 QQ 首页替换框架页面，无框架时和_self 相同）。

```
var  sName = '_blank'
```

```
window.open('http://www.qq.com',sName,'scrollbars=no,width=480,
height=420,left=75,top=20,status=no,resizable=yes');
```

- setInterval、clearInterval 方法以及 setTimeout、clearTimeout 方法均可实现定时器功能。前者可实现以指定的间隔时间（毫秒）重复执行某一功能操作；后者只能在指定的时间执行一次。例如 var id= setInterval(“code”,1000) 表示 1000 毫秒执行一次 code。Code 可以为一段 JavaScript 代码或一个 javascript 函数名。
- prompt、alert、confirm 方法实现对话框功能，其中 prompt 为接受用户输入字符串的对话框；alert 为输出文本对话框，confirm 实现具有确认和取消按钮的对话框。具体用法请看例 4.34。
- showModalDialog、showModelessDialog 方法用于从父窗口中弹出模态和无模态对话框。模态对话框是指只能用鼠标或键盘在该对话框中操作，而不能在弹出对话框的父窗口中进行任何操作。它们的用法和 open 方法类似，不过它们可以接受父窗口传递过来的参数。例如以下脚本实现了这样的功能：新打开一个居中显示的模态对话框，对话框左上角相对屏幕左上角的坐标位置为（75，20）像素，窗口宽 480、高 420 像素，在对话框加载 sample.htm 页面，父窗口的标题作为参数传递到对话框，对话框的边框呈 sunken 效果。在 sample.htm 中可用 window 对象的属性 window.dialogArguments 得到传递的参数值。

```
showModalDialog("sample.htm",document.title,"dialogWidth:480px;
dialogHeight:420px;
dialogLeft:75px;dialogTop:20px;center:yes;edge:sunken;");
```

- navigate 方法实现了类似超链接的功能，但只能在本窗口中打开另外一个 Web 页面。例如单击按钮后，从当前页面跳转为另外一个页面 sample2.htm：

```
<input type="button" value="Navigate"
onclick="javascript:window.navigate('sample2.htm');">
```

- status 属性可以在浏览器窗口状态栏显示给定的文本；screenLeft 和 screenTop 为浏览器窗口页面显示区域左上角相对屏幕左上角的横坐标和纵坐标；closed 属性可以返回窗口是否打开或者关闭的布尔值。
- window 对象的 frame 集合对象实现了在浏览器脚本程序中对框架的处理。frames（帧，又称框架）可以实现窗口的分隔操作。可以把一个窗口分割成多个部分，每个部分称为一个帧，每个帧本身已是一类窗口，继承了窗口对象所有的属性和方法。frames 集合对象是通过 HTML 标记<frame>、<frameset>来创建的，它包含了一个窗口中的全部帧数。用 parent 或 top 指定当前帧的父窗口，例如要得到窗口中所有帧对象集合，用 window.parent.frames 或 window.top.frames 即可。用 self 指定当前帧，例如指定当前帧跳转到另一个 Web 页，用 window.self.navigate("new.htm")即可。具体用法如下：

```
<script language="javascript">
var frm = window.parent.frames;
for (i=0; i < frm.length; i++)  alert(frm(i).name); //对每个帧进行循环
  alert(window.parent.frames.length); //显示一个窗口被分成了几个帧
var frm = document.frames;
//显示每个帧的 URL 地址
for (i=0; i < frm.length; i++) alert(frm(i).location);
window.self.navigate("new.htm"); //当前帧跳转到新的页
</script>
```

【例 4.34】 window 对象对话框演示。

```
<script language="JavaScript">
var test=window.prompt("请输入数据:");
var YorN=confirm("你输入的数据是"+test+", 确定吗? ");
if (YorN) alert("输入正确!");
else  alert("输入不正确!");
</script>
```

【例 4.35】 模态对话框演示。演示效果如图 4-5 所示。

```
<html><head><title>无标题页</title></head>
<SCRIPT language="JavaScript" >
function fnRandom(iModifier){
   return parseInt(Math.random()*iModifier); //形成随机数
}
function fnSetValues(){
   var iHeight=oForm.oHeight.options[oForm.oHeight.selectedIndex].text;
   if(iHeight.indexOf("Random")>-1){
      iHeight=fnRandom(document.body.clientHeight); /*随机数小于浏览器页面显示
区域的高度*/ }
   var sFeatures="dialogHeight: " + iHeight + "px;";
   return sFeatures;
}
function fnOpen(){
   var sFeatures="dialogWidth:200px;"+fnSetValues();
   showModalDialog("showModalDialog_target.htm", "", sFeatures)
}
</SCRIPT>
<body onmouseover="self.status='我是浏览器窗口状态栏,欢迎您!'"
 onload="alert('开始加载页面!') ">
<FORM NAME=oForm>
对话框高度<SELECT NAME="oHeight">
   <OPTION>-- Random --    </OPTION>
   <OPTION>120             </OPTION>
   <OPTION>200             </OPTION>
```

```
   <OPTION>250              </OPTION>
   <OPTION>300              </OPTION>
</SELECT><br>建立模态对话框
<INPUT TYPE="button" VALUE="Push To Create" onclick="fnOpen()">
</FORM>
</body> </html>
```

图 4-5　模态对话框演示

【例 4.36】 该例子实现了动态创建按钮和超链接 HTML 元素标记，并且用户双击鼠标后，浏览器页面开始往下滚动。运行此页面程序后注意将浏览器窗口收缩到足够小，以便出现滚动条，否则看不到效果。Span 标记是容器标记，用来放置新创建的 HTML 元素，也可用 Div 标记替代 Span 标记。本例中创建了一个按钮和超链接标记。window 对象有两个很重要的方法 window.setInterval() 和 window.clearInterval() 可以实现定时操作。window.scroll(x，y)方法可实现针对浏览器窗口页面显示区域左上角位置的屏幕滚动。

```
<html>
<head><title>无标题页</title>
</head>
<body>
<input type="button" value="click  me to add button" onclick="AddElement();" >
<SPAN ID="mySpan"></SPAN>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br>

<script language ="javascript">
function AddElement() {
    mySpan.innerHTML="";
       // 添加一个按钮标记
       //创建按钮元素
   var aElement=document.createElement("<input type='button'>");
   eval("aElement.value='请点击我'");  //指定按钮上的文字
   mySpan.appendChild(aElement);      //将建立的按钮放到 Span 容器标记中呈现出来
    aElement=document.createElement("A");          // 添加一个超链接标记
    aElement.innerText='我是超链接';    //指定超链接中的文字
```

```
    aElement.href="javascript:alert('A link.')";  //指定超链接的链接地址
    mySpan.appendChild(aElement);      //将建立的超链接放到 Span 容器标记中呈现出来
}

//双击鼠标滚动屏幕的代码
  var currentpos,timer;

  function initialize(){
    //每隔毫秒执行一次 scrollwindow 函数
    timer=setInterval ("scrollwindow()",30);
  }
  function sc(){
    clearInterval(timer); //定时器停止工作
  }
  function scrollwindow(){
   currentpos=document.body.scrollTop; //获取顶端的 y 坐标值（像素）
   window.scroll(0,++currentpos);      //沿 y 方向滚动一个像素单位
   if (currentpos !=document.body.scrollTop) sc();
  }
  document.onmousedown=sc;//调用鼠标单击事件处理函数，实现单击鼠标则停止屏幕滚动
  document.ondblclick=initialize;//调用鼠标双击事件处理函数，实现双击鼠标开始屏幕滚动
</script>
</body></html>
```

2．navigator 浏览器对象

navigator 对象是 window 的子对象。它提供浏览器名称、版本、客户端支持的 MIME 类型属性等环境信息。navigator 对象常用方法和属性如图 4-6 所示。

【例 4.37】 使用 navigator 对象获取浏览器相关信息。在浏览器中显示结果如图 4-7 所示。

window 对象 → navigator

方法	描述
javaEnabled()	判定浏览器是否可用 Java

属性	描述
appCodeName	返回浏览器代码名称
appMinorVersion	返回浏览器的简短说明
appName	返回浏览器的名称
appVersion	返回浏览器的版本信息
browserLanguage	返回当前浏览器使用的语言
cookieEnabled	判断浏览器是否支持 cookies
cpuClass	返回浏览器使用的 CPU 类型
onLine	判定系统是否处于离线模式
platform	返回系统的操作系统平台
systemLanguage	返回操作系统的默认语言

图 4-6　navigator 对象常用方法和属性

图 4-7　navigator 对象演示效果

```
<html><body>
<script type="text/javascript">
var browser=navigator.appName
var b_version=navigator.appVersion
var version=parseFloat(b_version)
var codeName=navigator.appCodeName
var cpu=navigator.cpuClass
document.write("浏览器名称: "+ browser)
document.write("<br />")
document.write("浏览器版本: "+ version)
document.write("<br />")
document.write("浏览器代码名称: "+ codeName)
document.write("<br />")
document.write("浏览器系统使用的 CPU 类型: "+ cpu)
</script>
</body></html>
```

3．位置对象（location）

location 对象是 window 的子对象。location 对象提供了与当前打开页面的 URL 一起工作的方法和属性。location 对象常用的方法和属性如图 4-8 所示。

方法	描述
assign("URL")	加载新的 Web 页面
reload()	重新载入当前文档，刷新页面
replace("URL")	用指定的新 web 页替换当前页面

属性	描述
hash	设置或返回 href 属性中在#符号后面的内容
host	设置或返回 URL 或本地所在的域名以及端口号
hostname	设置或返回本地或是 URL 所在的域名
href	设置或返回完整的 URL
pathname	设置或返回由 location 对象指定的 file 名称或是路径
port	设置或返与 URL 有关的端口号
protocol	设置或返回 URL 部分所使用的协议
search	设置或返回 href 属性里?号之后的内容

图 4-8　location 对象常用方法和属性

location 对象中 href 属性和 assign 和 replace 方法实现的功能与前述 window 对象的 navigate 方法相似，实现了在当前窗口中打开一个新的页面的功能。但用 replace 方法后，浏览器的前进、后退历史记录将丧失。请比较四者的用法：

window.location.href="http://www.qq.com";

window.location.assign("http://www.qq.com");

window.location.replace("http://www.qq.com");

window.navigate("http://www.qq.com");

当用户在程序中通过 href 或 url 传递参数的时候，例如 window.navigate ("http://sample.htm?x=5;y=6;z=7")语句中，传递了参数 x、y、z 的值，如何取到呢？用 location.search 就可以取到问号后的“x=5;y=6;z=7”字符串，然后再用 string 对象的 split 方法分解字符串后即可得到参数 x、y、z 的值。

【例 4.38】 显示当前页面的 URL，并刷新当前页面。

```
<html><head></head>
<body>
<input type="button" value="重新载入" onclick="alert(location.href);location.
reload();">
<SELECT  onchange="window.location.href=this.options[this.selectedIndex].
value">
```

```
<OPTION VALUE="http://www.microsoft.com/ie">Internet Explorer</OPTION>
<OPTION VALUE="http://www.microsoft.com">Microsoft Home</OPTION>
<OPTION VALUE="http://msdn.microsoft.com">Developer Network</OPTION>
</SELECT>
</body>
</html>
```

4．历史对象（history）

history 对象包含浏览器的浏览历史信息。history 对象的属性和方法如图 4-9 所示。用户在浏览器中通过点击超链接或其他方式不断跳转到新的页面，如果要后退看前面已经访问过的网页历史，可以在浏览器工具条单击“后退”。我们可以在程序中控制页面转向某一个网页历史记录。例如 history.go(-4)用来显示后退 4 步后的网页历史，history.go(4)用来显示前进 4 个页面后的网页历史。如果当前页面后的历史记录只有 2 步，虽然设定了前进 4 步，没有那么多，则显示最后一个网页历史。back 和 forward 方法的实现后退和前进一步的功能，等同于 history.go(-1)和 history.go(1)。

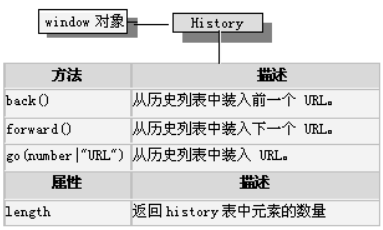

图 4-9　history 对象的属性和方法

【例 4.39】 使用 History 对象设置页面链接。

```
<html>
<head>
<title>history 对象示例</title>
</head>
<body>
<ul>
<li onclick="history.go(-1)">后退一页</li>
<li onclick="history.go(1)">前进一页</li>
</ul>
<a onClick="history.back()"><u>上一页</u></a>
<a onClick="history.forward()"><u>下一页</u></a>
</body>
</html>
```

5．事件对象（event）

event 对象是 window 对象的子对象，主要作用就是获取或者设置产生事件的对象是哪个对象、键盘按键的状态、当前鼠标指针的位置、鼠标按键的状态，等等。其常用的属性见表 4-6 所示。

表 4-6　event 事件对象的常用属性

属　　性	描述及例子
altKey	设置或获取 Alt 的按键状态值

altLeft	设置或获取左 Alt 的按键状态值
button	设置或获取用户按下鼠标键的方式。没按键取值 0；按左键取值 1；按右键取值 2；按下左右键取值 3；按下中间键取值 4
ClientX, clientY	设置或获取相对文档显示区域的鼠标指针坐标位置。
CtrlKey	设置或获取左 Ctrl 的按键状态值
CtrlLeft	设置或获取左左 Ctrl 的按键状态值
KeyCode	设置或获取产生按键事件的 Unicode 键盘代码
OffsetX, offsetY	设置或获取相对产生事件对象左上角的鼠标指针位置
returnValue	设置或获取事件的返回值
screenX ,screenY	获取相对用户屏幕左上角的鼠标指针坐标位置
shiftKey	设置或获取 Shift 的按键状态值
shiftLeft	设置或获取左 Shift 的按键状态值
srcElement	设置或获取产生事件的对象
type	设置或获取产生事件对象的事件名称
wheelDelta	设置或获取鼠标滚轮的方向和距离
x，y	设置或获取相对父元素左上角的鼠标指针坐标位置。例如鼠标在文本块标志 div 在移动时，event.x 和 event.y 将返回相对 Div 左上角的坐标位置

【例 4.40】 使用 srcElement 属性，判断鼠标单击了哪个元素。

```
<html><head><title>srcElement 演示</title></head>
<body  bgcolor=#FFFFCC >
<UL ID=oUL onclick="fnGetTags()" style="cursor:hand">
<LI>Item 1
   <UL>
   <LI>Sub Item 1.1
   <OL>
   <LI>Super Sub Item 1.1
   <LI>Super Sub Item 1.2
   </OL>
   <LI>Sub Item 1.2
   <LI>Sub Item 1.3
   </UL>
<LI>Item 2
   <UL>
   <LI>Sub Item 2.1
   <LI>Sub Item 2.3
   </UL>
<LI>Item 3
</UL>
<SCRIPT LANGUAGE="JavaScript">
function fnGetTags(){
   var oWorkItem=event.srcElement;     //获取被鼠标单击了的对象
   alert(oWorkItem.innerText); }       //显示该对象所包含的文本
</SCRIPT>
```

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

6．文档对象（document）

document 文档对象是浏览器对象的核心，主要作用就是把基本的 HTML 元素作为对象封装起来，提供给编程人员使用。编程人员利用这些对象，可以对 WWW 浏览器环境中的事件进行控制并作出处理。document 对象对实现 Web 页面信息交互起关键作用。document 对象所包含的属性、方法、事件、对象如图 4-10 所示。

对 document 对象的详尽介绍可参阅其他资料。下面介绍其常用的属性、方法、事件和对象。

（1）document 对象的常用属性见表 4-7 所示。

下面重点介绍 document 对象的 cookie 属性的使用。

cookie 是放在浏览器缓存中的一个文件，里面存放着各个参数名以及对应的参数值。cookie 中的参数是以分号相隔的，例如 "name=20;sex=male;color=red;expires=Sun May 27 22:04:25 UTC+0800 2008"。用户在打开同一个网站时，通过链接方式可能打开了多个浏览器窗口，这些窗口间需要共享信息时，cookie 就可以完成这项工作。cookie 存放的内容可以设置失效期限，既可以永久保留，也可以关闭网站后删除，也可以在指定时间内失效，通过 expires 指定 cookie 的失效日期，当没有失效日期时，关闭浏览器即失效。我们可以

图 4-10　document 对象

表 4-7　document 对象的常用属性

属　　性	描　　述
alinkColor	设置或获取文档中所有活动超链接对象的颜色（将鼠标在其上移动时）

vlinkColor	设置或获取文档中所有被访问过的超链接标记的颜色
linkColor	设置或获取文档中所有超链接标记的颜色
bgColor	设置或获取页面的背景颜色，例如背景颜色设为红色：document.bgColor ="red"
cookie	设置或获取 cookie
documentElement	获取 HTML 文档的根节点，例如：用 document.documentElement.innerHTML 可返回页面的源代码
fgColor	设置或获取页面的前景色
fileCreatedDate	获取 HTML 文档创建的日期
fileModifiedDate	获取 HTML 文档最后修改日期。例如：alert(document.fileModifiedDate);
fileSize	获取 HTML 文档所占磁盘空间大小
lastModified	取 HTML 文档最后修改日期和时间
parentWindow	该文档的父窗口
protocol	设置或获取 URL 地址中协议部分内容
referrer	假定当前页是从另外一个页面的超链接跳转的，该属性可以获取上一个页面的 URL 地址
uniqueID	获取为对象自动生成的一个唯一标识号
URL	设置或获取当前文档的 URL 地址。例如 document.URL="11.htm" 可实现页面的跳转
URLUnencoded	将 URL 地址中的特殊字符代码变成 ASCII 码。例如 alert(document.URL)返回"file://C:\Documents%20and%20Settings\aa.htm",alert(document.URLUnencoded)将返回"file://C:\Documents and Settings\aa.htm"

保存用户输入的参数到 cookie 中，以后可以恢复显示。例如在登录一个系统时，需要用户从学生、科任教师、班主任、辅导员中选择一种用户类型登录，通过 cookie 保存选定的用户类型后，以后再次登录系统，就不要再让用户选择用户类型，可从 cookie 中取出设定的用户类型作为默认值。

【例 4.41】 设置 cookie 和获取 cookie 的例子。该例子有两个通用的函数 setCookie 和 getCookie 分别设置和获取 cookie。用户在文本框中输入姓名，单击按钮“姓名保存到 cookie”后，再单击“从 cookie 中得到姓名”按钮，在第二个文本框中显示取到的姓名。

```
<HTML><HEAD><TITLE>First Document</TITLE>
<script type="text/javascript">
function getCookie(sName)  //从 cookie 中获取参数 name 的值
{  // cookie 中的参数是以分号相隔的，例如"name=20;sex=male;color=red;"
  var aCookie = document.cookie.split("; ");
  for (var i=0; i < aCookie.length; i++)
  {// 对存放在数组 aCookie 中的每一个"参数名=参数值"进行循环，找到要获取参数值的参数名
    var aCrumb = aCookie[i].split("=");
    if (sName == aCrumb[0])
      return unescape(aCrumb[1]); //如果找到则返回参数值
  }
  return null; // cookie 中请求的参数名不存在时返回 null
}

// name: 参数, value: 参数值, expires: 失效日期,
//功能：将参数 name 的值 value 和失效日期 expires 写入一个 cookie 中
function setCookie(name, value, expires)
```

```
{   var expStr    = ( (expires == null) ? "" : ("; expires=" + expires) );
   window.document.cookie = name + "=" + escape(value) + expStr;
}
</script>
</HEAD>
<BODY bgcolor=#FFFFCC>
<Input id="yourName" type ="text" value = "Tim">
<Input type ="button" value = "姓名保存到 cookie" onclick="setCookie('name',
yourName. value,'Sun May 27 22:04:25 UTC+0800 2008');">
<Input id="GetName" type ="text" value = "">
<Input type ="button" value = "从 cookie 中得到姓名" onclick="GetName.value=
getCookie('name');">
</BODY></html>
```

（2）document 对象的常用方法见表 4-8 所示。

文档对象的方法 write 和 writeln 方法主要用来实现在 Web 页面上显示输出信息。在实际使用中，writeln()与 write()唯一不同之处在于在尾部加了一个换行符。 open 方法有两种用法：

- 与 window.open()用法完全一致，用来创建一个新的窗口或在指定的命令窗口内打开文档；

表 4-8　document 对象的常用方法

方　法	描　述
attachEvent	可以为某个对象动态绑定一个事件处理程序
write	在当前页面中输出 HTML 形式的文本串
writeln	在当前页面中输出 HTML 形式的文本串，文本串末尾追加一个换行键
close	和 write 或 writeln 配对使用，关闭输出 HTML 文本流，立即在浏览器中显示
createAttribute	动态创建一个指定属性名的属性
createElement	创建一个指定标签的元素实例
createStyleSheet	为文档创建样式单
detachEvent	将事件与对象分离，对象的事件不再起作用
elementFromPoint	获取放在 x，y 坐标处的 HTML 元素对象
execCommand	在当前文档中执行命令
focus	使元素对象得到焦点，执行 onfocus 事件
getElementById	通过 Id 属性返回第一个元素对象
getElementsByName	通过 NAME 属性返回元素对象集合
getElementsByTagName	通过标记元素名称返回元素对象集合
hasFocus	判断当前对象是否得到焦点
open	当只指定 url 和 name 参数时，在当前文档中输出 HTML 文本流，用 close 关闭；当指定其他附加参数时，其作用与 window.open 相同
attachEvent	可以为某个对象动态绑定一个事件处理程序

- 在一个已存在文件中写入内容或创建一个新文件来写入内容。在完成对 Web 文档的写操作后，要使用 close()方法来实现对输出流的关闭。在使用 open()来打开一个新流时，可为文档指定一个有效的文档类型，有效文档类型包括 text/html、text/gif、text/xim、text/plugin 等。

【例 4.42】 文档对象 write、open、close 方法示例。单击 Finish Sentence 按钮原地输出另外一个页面。单击“打开新窗口”按钮打开 QQ 首页。初始效果如图 4-11 所示。

图 4-11 页面初始显示效果

```
<HTML><HEAD><TITLE>First Document</TITLE>
<script type="text/javascript">
function replace(){
    var oNewDoc = document.open("text/html", "replace");
    var sMarkup = "<HTML><HEAD><TITLE>New Document</TITLE><BODY>Hello, world
</BODY></HTML>";
    oNewDoc.write(sMarkup);
oNewDoc.close();
}
function openwin() {
```

```
document.open('http://www.qq.com','_blank','scrollbars=no,width=480,heigh
t=420,left=75,top=20,status=no,resizable=yes');  //新打开一个窗口 }
</script>
</HEAD>
<BODY bgcolor=#FFFFCC>
<h4>I just want to say</h4><br>
<!--Button will call the replace function and replace the current page with
a new one-->
<Input type ="button" value = "Finish Sentence" onclick="replace();">

<script type="text/javascript">
//在当前页面显示一个图片和一个 cancel 按钮
document.writeln("<p></p><hr><img src='web.gif'>");
document.writeln("<Input type ='button' value ='cancel'>");
</script>
<Input type ="button" value ="打开新窗口" onclick="openwin();">
</BODY>
</html>
```

（3）document 对象的事件。

文档对象的事件除了响应键盘、鼠标常规操作事件外，还增加了其他大量事件，例如鼠标的拖拉操作事件（ondrag、ondragend、ondragenter、ondragleave、ondragover、ondragstart、

ondrop）、快捷菜单事件 oncontextmenu 等。例如以下代码实现了在浏览器窗口文本块 span 区域，按下鼠标右键时将不出现快捷菜单。

```
<SPAN STYLE="width:300; background-color:blue; color:white;" oncontextmenu=
"return false">
<P>The context menu never displays when you right-click in this box.</P>
</SPAN>
```

（4）document 的对象。

document 所包含的对象主要是集合对象，包括 all、anchors、applets、childNodes、embeds、forms、frames、images、links、namespaces、scripts、styleSheets 等。例如以下代码演示了 all 的用法：

```
var oItem = document.all; //得到页面中的所有对象，并显示每个对象对应的标记名称
if (oItem!=null)  for (i=0; i<oItem.length; i++) alert(oItem.item(i).tagName);
//得到 id 为 Sample 的所有元素并显示出标记名称
var oItem = document.all.item("Sample");
If (oItem != null) for (i=0; i<oItem.length; i++) alert(oItem.item(i).tagName);
```

要详细介绍各个对象的使用超出了本书范围。为方便读者查阅，表 4-9 列出了 DHTML 常用对象名称。

表 4-9　DHTML 常用对象名称

对　　象	描　　述
window	当出现<body>或是<frameset>标签，这个对象就会自动建立起来
location	含有当前 URL 的信息
screen	由脚本工作环境引擎自动建立，它包含了有关客户显示屏幕的信息
event	代表事件的状态，比如哪个元素的事件发生了；键盘按键的状态；鼠标的位置；鼠标按钮的状态等
navigator	获取客户端信息，包括浏览器类型、操作系统类型、CPU 类型等
history	通过 window 对象的 history 属性所访问的预先确定对象。这个对象由一组 URLs 所构成。这些 URLs 是所有用户曾经在浏览器中访问过的 URLs
document	可用来访问所有在页面中的元素
anchor	代表了 HTML 的 a 元素（超链接）
applet	代表了 HTML 的 applet 元素，applet 元素可用来放置页面内可执行的内容
area	代表了图像映射区域。
base	代表了 HTML 的 base 元素
basefont	HTML 的 basefont 元素
body	文档的主体（body）
button	HTML 表单上的按钮。HTML 表单中只要出现了<input type="button">标签，一个 button 对象就建立了
checkbox	HTML 表单中的复选框。只要 HTML 表单中出现了<input type="checkbox">标签，就会建立起 checkbox 对象

fileupload	当 HTML 表单中有<input type="file">,FileUpload 对象就建立起来了
form	表单可用于用户信息的输入递交。可代表 HTML 中的 form 元素
frame	代表了 HTML 中的框架
frameset	HTML 的框架集
hidden	在 HTML 中的隐藏区域。每当 HTML 表单中出现<input type="hidden">标签，hidden 对象就建立起来了
iframe	HTML 中的内联框架
image	HTML 的 img 元素
link	HTML 的 link 元素。只能在<head>标签里使用 link 元素
meta	HTML 的 meta 元素
option	在 HTML 表单中的选择项。每当 HTML 表单中出现<option>标签，option 对象就建立了
password	代表了 HTML 表单中的 password 区域。每当 HTML 表单中出现<input type="password">标签，就会建立该对象
radio	代表了 HTML 表单中的单选按钮。每当表单中出现了<input type="radio">标签，就会建立该对象
reset	代表了 HTML 表单中的 reset 按钮。每当表单出现了<input type="reset"> 标签就会建立该对象
select	代表了 HTML 表单中的选择列表。每当表单中出现<select>标签就会建立该对象
style	代表了独立的样式声明。可以从文档或使用样式的元素中访问这个对象
submit	代表了 HTML 表单中的提交按钮。每当表单中出现<input type="submit"> 标签该对象就建立起来了

续表

对　象	描　述
table	代表了 HTML 的表格元素
tabledata	代表了 HTML 里的 td 元素
tableheader	代表了 HTML 里的 th 元素
tablerow	代表了 HTML 里的 tr 元素
text	代表了表单中的文字输入区域。每当表单中出现<input type="text">标签该对象就建立起来了
textarea	代表了 HTML 的 textarea 元素

下面通过大量典型的实例帮助读者掌握其编程方法。

【例 4.43】 使用 documentElement 将页面源代码调入编辑框；用滚动鼠标滚轮的方法放大或缩小图片。

```
<html><head><title>将页面源代码调入编辑框</title></head>
<body>
<SCRIPT type="text/javascript">
function fnGetHTML(){
   var sData = document.documentElement.innerHTML;
   oResults.value=sData;
}
```

```
function zoomimg(o){                          //用滚动鼠标滚轮的方法放大或缩小图片
     var zoom=parseInt(o.style.zoom, 10)||100;
zoom+=event.wheelDelta/12;                    //滚轮滚过的距离
     if (zoom>0) o.style.zoom=zoom+'%';       //设置缩放比例
    return false;
}
</SCRIPT>
<A  href="javascript:fnGetHTML()">将页面源代码调入编辑框</A>
<TEXTAREA ID = oResults COLS = 50 ROWS = 10></TEXTAREA>

<a href="web.gif" onmousewheel="return zoomimg(this)" target="_blank">
<img src="web.gif" border=1 title="点击可看原图"
onload="javascript:if(this.width>740)this.width=740;"
   onmouseover="javascript:if(this.width>740)this.width=740;"></a>
</body>
</html>
```

【例 4.44】 使用 document 对象查看元素名称。

```
<html><head>
<script type="text/javascript">
function getElement(){
var x=document.getElementById("myHeader")
alert("这是一个" + x.tagName + " 元素")
}
</script>
</head><body>
<h3 id="myHeader" onclick="getElement()">点击查看此元素名称</h3>
</body>
</html>
```

【例 4.45】 使用 Form 数组和 Form 名称使得两个 Form 中的文本输入内容保持一致。

```
<html><head><title>form 对象</title></head>
<body>
<form>
<input type="text" onChange="document.my.elements[0].value=this.value;" >
</form>
<form name="my">
<input type="text" onChange="document.forms[0].elements[0].value=this.value;">
</form>
</body></html>
```

在浏览器中显示结果如图 4-12 所示。

【例 4.46】 文档对象应用举例。

```
<html><head></head>
<body>
<form Name="mytable">请输入数据:
<Input Type="text" Name="text1" value="">
</form>
<A name="Link1" href="3-1.html">链接到第一个文本</a><br>
<A name="Link2" href="3-2.html">链接到第二个文本</a><br>
<A name="Link2" href="3-3.html">链接到第三个文本</a><br>
<A href="#Link1">第一锚点</a>
<A href="#Link2">第二锚点</a>
<A Href="#Link3">第三锚点</a><br>
<Script Language="javascript">
document.write("文档有"+document.links.length+"个链接"+"<br>");
document.write("文档有"+document.anchors.length+"个锚点"+"<br>");
document.write("文档有"+document.forms.length+"个窗体");
</script>
</body></html>
```

在浏览器中显示结果如图 4-13 所示。

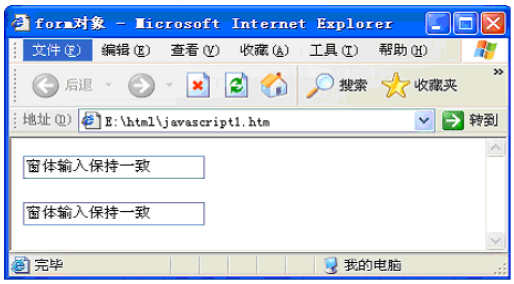

图 4-12　Form 对象示例

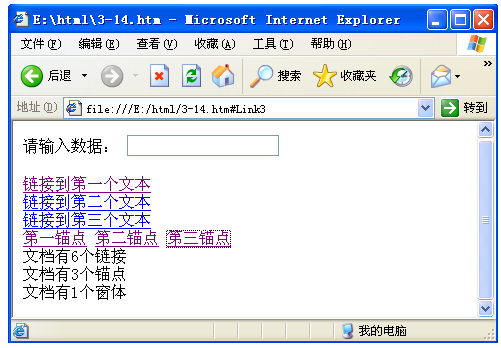

图 4-13　document 对象应用示例

【例 4.47】 设计一个输入个人信息的页面示例。源代码如下：

```
<html>
<head><script language="JavaScript">
sex=new Array();
sex[0]="Male";
sex[1]="Female";
sele=0;
sex_sele=0;
function VertifyAndChgText(){
  var Length=document.forms[0].length;
```

```
  var Type,Empty=false;
  for(var i=0;i<Length;i++){
  Type=document.forms[0].elements[i].type;
if (Type=="text")
    if (document.forms[0].elements[i].value=="")
      empty=true;
  }
  if (!Empty){
    name="您的名字是"+document.forms[0].NameText.value+"\n";
    alias="您的别名是"+document.forms[0].AliasText.value+"\n";
    sex_1="您的性别是"+sex[sex_sele]+"\n";
    area="您所在的地区是"+document.forms[0].area.options[sele].text+"\n";
    exp="备注信息是"+document.forms[0].exp.value+"\n";
    document.forms[0].info.value=name+alias+sex_1+area+exp;
  }
  else   alert("您的信息尚未完全输入!");
  return Empty;
}
</script>
</head>
<body>
<h3 align=center>请输入您的个人信息</h3>
<form>
您的姓名: <input type=text name="NameText" size=15><br>
您的别名: <input type=text name="AliasText" size=15><br>
您的性别: <input type="radio" name="sex" onClick="sex_sele=0">Male
<input type="radio" name="sex" onClick="sex_sele=1">Female<br>
你所在地区: <select name="area" onChange="sele=this.selectedIndex">
<option value="1" selected >云南   </option>
<option value="2">贵州             </option>
<option value="3">四川             </option>
<option value="4">西藏             </option>
<option value="5">重庆</option></select><br>
备注: <textarea name="exp" rows=4 cols=25></textarea><br><br><br>
<input type=button value="提交信息" onClick="VertifyAndChgText()">
<br>
您已输入的信息是: <textarea name="info" rows=6 cols=30></textarea><r><br>
</form></body></html>
```

在浏览器中显示结果如图 4-14 所示。

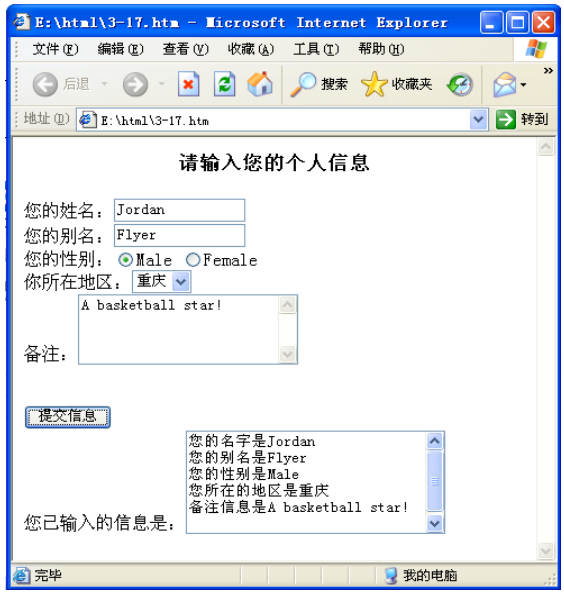

图 4-14　Form 对象综合应用示例

【例 4.48】 逐条显示信息的跑马灯。源代码如下：

```
<html>
<head></head>
<script language="javascript">
<!--
  function makearray(size){
```

```
   this.length=size;
   for(i=1;i<=size;i++){this[i]=0}
   return this;
  }
 msg=new makearray(2);
  msg[1]="嗨！您好。";
  msg[2]="可根据需要显示不同内容！";
  interval = 100;
  seq = 0;
  i=1;
function Scroll(){
    document.myForm.myText.value = msg[i].substring(0, seq+1);
    seq++;
  if ( seq >= msg[i].length ) { seq = 0 ;i++;interval=900};
    if(i>2){i=1};
    window.setTimeout("Scroll();", interval );
interval=100;
}
```

```
-->
</script>
<body OnLoad="Scroll()">
<form name="myForm">
<input type="text" name="myText" size="50">
</form></body></html>
```

【例 4.49】 先出现文件上载控件，没有“提交”按钮，在检查上传图片的大小后，若图片符合要求则动态创建一个“提交”按钮。源代码如下：

```
<html>
<head>
<title>图片上传</title>
<script type="text/javascript">
   function WH_Show(f){
   obj=document.createElement("IMG");
   obj.src=f.value;
   if(obj.width>640 || obj.height>480){
      alert("图片太大，请上传小于 x480 的图片。")
   }
   else{
  document.getElementById('sub').innerHTML='<INPUT TYPE="submit" NAME="Submit"
VALUE="提交">'
      }
   }
</script>
```

```
</head>
<body>
<FORM NAME="form1" METHOD="post" ACTION="upfile.aspx" ENCTYPE="multipart/
form-data" >
<INPUT TYPE="hidden" size="40" NAME="act" VALUE="upload">
<input type="file" onchange="WH_Show(this)">
<div ID="sub"></div>
</FORM></body></html>
```

【例 4.50】 实现鼠标悬停时，显示放大的图片。

```
<html>
<head>
<title>用 HTML DOM 放大图片</title>
</head>
<body>
<BR>原图: <BR><img src=img1.JPG onmousemove="zoom()" id=srcImg>
```

```
<BR>局部放大图: <BR><div style="overflow:hidden"><img id=zoomImg></div>
<script language="JavaScript">
<!--
zoomImg.src = srcImg.src;
srcImg.height = srcImg.height/4;
var zoomRate = 5;
zoomImg.height = srcImg.height*zoomRate;
zoomImg.parentNode.style.width = srcImg.width;
zoomImg.parentNode.style.height = srcImg.height;
function zoom(){
var elm = event.srcElement;
h = elm.offsetHeight/zoomRate/2;
w = elm.offsetWidth/zoomRate/2;
var x = event.x-elm.offsetLeft;
x=x<(elm.offsetWidth-w)?x<w?w:x:elm.offsetWidth-w;
zoomImg.style.marginLeft=(w-x)*zoomRate;
var y = event.y-elm.offsetTop;
y=y<(elm.offsetHeight-h)?y<h?h:y:elm.offsetHeight-h;
zoomImg.style.marginTop=(h-y)*zoomRate;
}
-->
</script>
</body>
</html>
```

在浏览器中显示结果如图 4-15 所示。

【例 4.51】 动态生成一个表格。源程序如下：

```
<html><head><title>使用 DOM 生成一张表格</title>
<script language="JavaScript">
function genTable(pNode){
var i,j;
var contents=new Array(3);
for (i=0;i<3;i++)  contents[i]=new Array(3);
contents[0][0]="书名";
contents[0][1]="出版社";
contents[1][0]="C++程序设计教程";
contents[1][1]="清华大学出版社";
contents[2][0]="Web 程序设计";
contents[2][1]="电子工业出版社";
var tableNode=document.createElement("TABLE");
var tBodyNode=document.createElement("TBODY");
```

```
var t1,t2;
for (i=0;i<3;i++){
  t1=document.createElement("TR");
    tBodyNode.appendChild(t1);
    for (j=0;j<2;j++){
      t1=document.createElement("TD");
      t2=document.createTextNode(contents[i][j]);
      t1.appendChild(t2);
      tBodyNode.childNodes[i].appendChild(t1);
    }
}
pNode.appendChild(tableNode);
tableNode.id="test";
tableNode.border=2;
tableNode.appendChild(tBodyNode);
}
</script>
</head>
<body id="tableTest">
<h2 onClick="genTable(tableTest)">单击此处将生成一个表格</h2>
<hr></body></html>
```

在浏览器中显示结果如图 4-16 所示。

图 4-15　用 HTML DOM 放大图片

图 4-16　使用 DOM 生成一张表格

【例 4.52】 用鼠标器滚轮放大和缩小图形。

```
<script language="javascript">
var count = 10;
function resizeimg(oImage) {
count = Counting(count);
Resize(oImage,count);
```

```
return false;
}
function Counting(newzoom){
if (event.wheelDelta >= 120)
newzoom++; //鼠标滚轮滚动 120 个单位距离，就放大或缩小 1%
else if (event.wheelDelta <= -120)
newzoom--;
if (newzoom<2) newzoom=2;              //只允许缩小到 20%
if (newzoom>50) newzoom=50;            //只允许放大到 500%
return newzoom;
}
function Resize(oImage,newzoom){
oImage.style.zoom = newzoom + '0%';    //设置图片的缩放属性
count=newzoom;
}
</script>
<img  onDblClick='return  Resize(this,10);return  false;'    onmousewheel=
'return resizeimg(this)' border="0" src="upload/upfile/map/cjtj/syt2.gif"
width="500" height="369">
```

【例 4.53】 播放 WMA 或 MP3 等格式的媒体文件。

```
<html>
<script language="JScript">
function CreateControl(DivID, CLSID, ObjectID,WIDTH, HEIGHT, URL, AUTOSTART){
  var d = document.getElementById(DivID);
d.innerHTML='<object  classid='+CLSID+'  id='+ObjectID+'  width='+WIDTH+'
height='+HEIGHT+'>
  <param name="URL" value=' + URL + '>  <param name="autoStart" value=' +
AUTOSTART + '/>';
}
</script>
 <head></head>
 <body>
   <div id="EXAMPLE_DIV_ID">  This text will be replaced by the control </div>
   <script language="JScript">
CreateControl("EXAMPLE_DIV_ID","clsid:6BF52A52-394A-11d3-B153-00C04F79FAA6",
"EXAMPLE_OBJECT_ID", "300", "200", "file:///C:/TDdownload/红旗飘飘.wma","-1");
   </script>
 </body>
</html>

<!--  以下是另外一种播放方法-->
<html>
  <body>
```

```
    <div id="DivID">
 <script language="javascript">
       var myObject = document.createElement('object');
DivID.appendChild(myObject);
myObject.width = "200";
myObject.height = "100";
myObject.classid= "clsid:6BF52A52-394A-11d3-B153-00C04F79FAA6";
myObject.URL = "file:///C:/TDdownload/红旗飘飘.wma";
myObject.uiMode = "none" ;
</script>
</body>
</html>
```

4.2.4　HTML DOM 树简介

HTML DOM 是一种结构化的对象模型，采用 DOM 技术访问和更新 HTML 页面内容时，首先依据 HTML 源代码，建立页面的树型结构模型，然后按照树型结构的层次关系来操纵 Web 页面。图 4-17 所示为一个 Web 页文件所对应的 DOM 节点树，又称文档大纲。

在 DOM 树型结构中，每个节点都是一个对象，各节点对象都有属性和方法。

DOM 有两个对象集合：attributes 和 chileNodes。attributes 是节点属性的对象集合。chileNodes 是子节点的对象集合，使用从 0 开始的索引值进行访问。

DOM 树型结构节点有只读属性和读写属性两类。通过只读属性可以浏览节点，并可获得节点的类型及名称等信息；通过读写属性可以访问文字节点的内容。DOM 树节点的属性如表 4-10 所示。

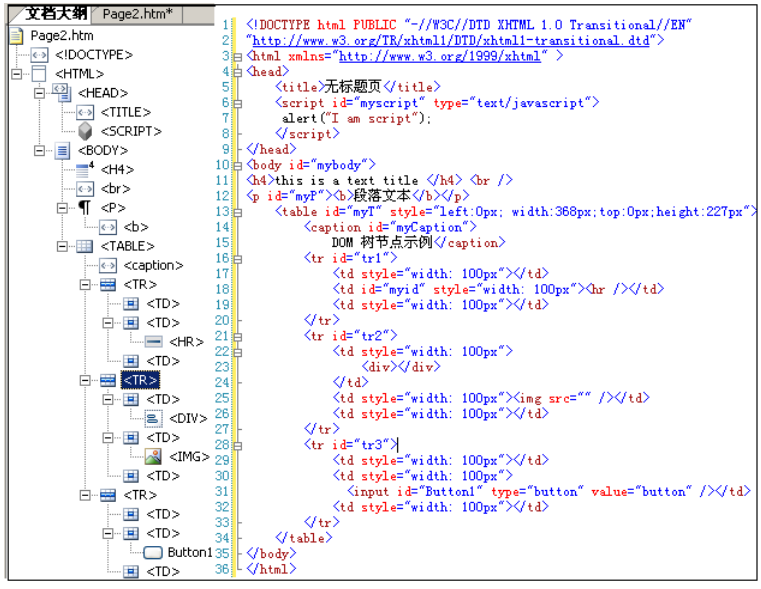

图 4-17　文档大纲

表 4-10　DOM 树节点的属性

属　性	访　问	说　明
nodeName	只读	返回节点的标记名
nodeType	只读	返回节点的类型：1-标记；2-树型；3-文字节点
firstChild	只读	返回第一个子节点的对象集合
lastChild	只读	返回最后一个子节点的对象集合
parentNode	只读	返回父节点对象
previousSibling	只读	返回左兄弟节点对象
nextSibling	只读	返回有兄弟节点对象
data	读写	文字节点的内容，其他节点返回 undefined
nodeValue	读写	文字节点的内容，其他节点返回 null

用 DOM 方法可以动态创建 HTML 文档或 HTML 元素，并可以通过 JavaScript 程序随时改变文档的节点结构或内容，建立动态网页效果。DOM 树的常用方法如表 4-11 所示。

表 4-11　DOM 树的常用方法

方法及语法	说　明
objParent.appendChild(objChild)	为 objParent 添加子节点 objChild，返回新增节点对象
objChild.appendElement(objParent)	将 objChild 新增为 objParent 的子节点
objNode.SetAttribute(sName, vValue)	设置 objNode 的属性名和属性值
objNode.clearAttributes( )	清除 objNode 的所有属性
document.createElement (String)	建立一个 HTML 节点对象，参数 TagName 为标记的名称
objNode.cloneNode (deep)	复制节点 objNode，若 deep 为 false，则只复制该节点；否则，复制以该节点为根的整个树

续表

方法及语法	说　明
objNode.hasChildNodes( )	判断 objNode 是否有子节点，若有则返回 true，否则返回 false
objParent.insertBefore (objChild,objBrother)	在节点 objParent 的子节点 objBrother 之前插入一个新的子节点 objChild
ObjTarget.mergeAttributes(objBrother)	将节点 objSource 的所有属性复制到节点 objTarget 中
objNode.removeNode(deep)	删除节点 objNode，若 deep 为 false，则只删除该节点；否则，删除以该节点为根的子树
objNode. replaceNode(objNew)	用节点 objNew 替换节点 objNode
objNode1.swapNode(objNode2)	交换节点 objNode1 与 objNode2

在用 createElement 方法创建一个元素时，其中的参数可以使用表 4-12 中所列 DHTML 对象的名称。对象名称的含义和标记名称基本相同，具体说明可参阅相关资料。

表 4-12　DHTML 对象的名称

A、acronym、address、applet、area、attribute、b、base、baseFont、bgSound、big、blockQuote、body、br、button、caption、center、cite、clientInformation、clipboardData、code、col、colGroup、comment、currentStyle、custom、dataTransfer、dd、defaults、del、dfn、dir、div、dl、document、dt、em、embed、

event、external、fieldSet、font、font、form、form、frame、frame、frameSet、frameSet、FRAMESET elements.、head、history、hn、hr、html、HTML Comment、i、iframe、img、implementation、IMPORT、input、input type=button、input type=checkbox、input type=file、input type=hidden、input type=image、input type=password、input type=radio、input type=reset、input type=submit、input type=text、ins、isIndex、kbd、label、legend、li、link、listing、location、map、marquee、menu、meta、namespace、navigator、nextID、noBR、noFrames、noScript、object、ol、optGroup、option、P、page、param、plainText、popup、pre、Q、rt、rule、runtimeStyle、s、samp、screen、script、select、selection、small、span、strike、strong、style、style、styleSheet、sub、sup、table、tBody、td、textArea、TextNode、TextRange、TextRectangle、tFoot、tHead、title、tr、tt、u、ul、userProfile、var、wbr、window、xml

例如创建一个按钮元素：document.createElement("<input type='button'>")；创建一个超链接元素：document.createElement("<A>")。下面的示例代码给出了 DOM 树的编程方法。

【例 4.54】 请读者体验以下 DOM 树编程过程。单击按钮 button 后的执行结果如图 4-18 所示。

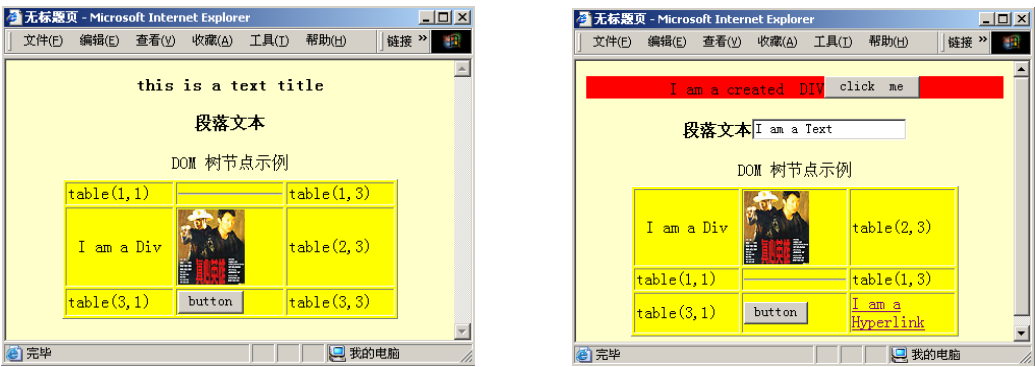

图 4-18　DOM 树编程执行效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>无标题页</title>
</head>
<body id="mybody" bgcolor=#FFFFCC>
<h4 id="myH" align="center" >this is a text title </h4>
<p id="myP" align ="center"><b>段落文本</b></p>
    <table id="myT"  bgcolor="yellow" align="center" border="1" >
        <caption id="myCaption">
            DOM 树节点示例</caption>
        <tr id="tr1">
            <td style="width: 100px">table(1,1)</td>
            <td id="myid" style="width: 100px"><hr /></td>
            <td style="width: 100px">table(1,3)</td>
        </tr>
```

```
      <tr id="tr2">
         <td style="width: 100px">
            <div  ID="oDiv"  align="center" onclick="alert('click');"
  onmouseover="this.style.color='#0000FF';"
              onmouseout="this.style.color='#FF0000';">I am a Div</div>
         </td>
         <td style="width: 100px"><img src="ad1.jpg" /></td>
         <td style="width: 100px">table(2,3)</td>
      </tr>
      <tr id="tr3">
        <td style="width: 100px">table(3,1) </td>
        <td style="width: 100px">
          <input id="Button1" type="button" value="button" onclick="dom_
example();" /></td>
        <td style="width: 100px">table(3,3)</td>
      </tr>
   </table>
 <script type="text/javascript">
 function fun_click() {  //定义一个事件函数
   document.getElementById("mybutton").style.backgroundColor="blue";
   alert("yes, that's me");
 }
  function dom_example() {
  //1.在段落文本 id 为 myP 处添加一个 text 文本框标记，设置 Value 属性为“I am a Text”
 var nod1=document.createElement("<input type='text'>");
 nod1.setAttribute("value","I am a Text");
 myP.insertBefore(nod1);

 //2. 在文档开始处添加一个背景为红色的 Div。id 属性设为 mydiv,里面添加文字"I am a
 //created  DIV"
 var nod2=document.createElement("<div style='background-color:red;'>");
 nod2.setAttribute("id","mydiv");
 document.body.insertBefore(nod2);
 mydiv.innerText="I am a created  DIV";
 //创建一个按钮
 var  nod3=document.createElement("<input  type='button'  value='click  me'
id='mybutton'>");
 var mybutton=mydiv.appendChild(nod3);
 //为该按钮添加一个 click 事件处理函数
 mybutton.attachEvent("onclick",fun_click);
//3.对 ID 为 myT 的 table 所设置的所有属性进行循环，读取并显示出来
  for(var i=0;i<myT.attributes.length;i++){
       if(myT.attributes[i].specified){
```

```
          alert(myT.attributes[i].nodeName+"="+myT.attributes[i].nodeValue);
        }
   }

//4.将表格 id 为 tr1 的第行和 id 为 tr2 的第行进行交换
  tr1.swapNode(tr2);

//5. 将 id 为 myH 的标题文本换成另外一个节点(前面创建的 nod2 或 mydiv)
  myH.replaceNode(nod2);

//6.将 oDiv 的属性复制给第二步中创建的元素 mydiv，但 ID 属性不复制。将鼠标在 mydiv 上移
//动将会发现事件属性也被复制
mydiv. mergeAttributes(oDiv,true);
// 清除 mydiv 的属性可用 mydiv.clearAttributes(); 但不清除事件属性

//7.对于表格第三行的 tr3 来说，有三个单元格，即有三个子节点
  if( tr3.hasChildNodes()){
       //返回头节点(第一个单元格)中的内容
       alert('firstChild='+tr3.firstChild.innerHTML);
      //返回最后一个节点(第三个单元格)中的内容
      alert('lastChild='+tr3.lastChild.innerHTML);
     for (i=0; i<tr3.childNodes.length; i++) {  //对所有子节点循环
       var myrow=tr3.childNodes(i); //得到每个子节点对象，即每个单元格对象
       alert(myrow.innerHTML); //显示每个单元格中的标记元素信息
       //返回同胞节点（单元格）中的内容
       if(i==0)alert(myrow.nextSibling.innerHTML);
//设置最后一个单元格的内容为一个超链接
if(i==tr3.childNodes.length-1)
myrow.innerHTML="<A href='#'>I am a Hyperlink </A>";
     }
   }
}
</script>
</body>
</html>
```

4.3　如何用 jQuery 简化 JavaScript 开发

4.3.1　jQuery 简介

jQuery 是继 prototype 之后的又一个优秀的 JavaScript 框架。它是由 John Resig 于 2006 年初创建的，它凭借简洁的语法和跨平台的兼容性，大大简化了 JavaScript 开发人员遍历 HTML 文档、操作 DOM、处理事件、执行动画和开发 Ajax 的操作。其独特而优雅的代码

风格改变了 JavaScript 程序员的设计思路和编写程序的方式。

jQuery 是一个轻量级的脚本，其代码简练、语义易懂，支持 CSS1-CSS3 以及基本的 xPath，能将 JavaScript 代码和 HTML 代码完全分离，便于代码的维护和修改，且可很容易为 jQuery 扩展其他功能，jQuery 主要可实现如下功能：

- **获取文档中的元素**。jQuery 为准确地获取需要检查或操作的文档元素，提供了可靠而富有效率的选择符机制。
- **修改页面外观**。jQuery 提供了跨浏览器的标准解决方案，即使在页面呈现以后，仍能改变文档中某个部分的类或者个别的样式属性。
- **改变文档的内容**。jQuery 能够影响的范围并不局限于简单的外观变化，使用少量的代码，jQuery 就能改变文档的内容。
- **响应用户的交互操作**。jQuery 提供了形形色色的页面事件的适当方式，而不需要使用事件处理程序使 HTML 代码看起来杂乱。此外，它的事件处理 API 也消除了经常困扰 Web 开发人员的浏览器不一致性问题。
- **为页面添加动态效果**。为了实现某种交互行为，设计者必须向用户提供视觉上的反馈。jQuery 内置的一批淡入、擦除之类的效果，以及制作新效果的工具包，为此提供了便利。
- **无需刷新页面从服务器获取信息**。这种编程模式就是众所周知的 AJAX，它能帮助 Web 开发人员创建出反应敏感、功能丰富的网站。jQuery 通过消除这一过程中的浏览器特定的复杂性，使开发人员得以专注于服务器的功能设计。
- **简化常用的 JavaScript 任务**。除了这些完全针对文档的特性之外，jQuery 也提供了对基本的 JavaScript 结构（例如迭代和数组操作等）的增强。

jQuery 是作为一个 js 文件来分布的，可以从 jQuery 的官方网站（http://jquery.com/）下载最新的 jQuery 库文件。jQuery 库有三个版本：压缩版、开发版和 Visual Studio 文档版。其中开发版是标准的 JavaScript 文件，具有注释、空格和换行。压缩版是将标准版中的所有空白和注释删除以后得到的版本，这个版本文件更小，下载速度更快，能够提高网页加载速度。

使用 jQuery 库不需要安装，只需在相关的 HTML 文档中简单地引用该库文件的位置。方法与引用其它 JavaScript 文件相同，使用 script 标记通过 src 属性指定要加载的文件，例如将 jquery-1.6.2.js 放在目录 scripts 下，在 HTML 文档中的代码如下所示：

```
<script src="../Script/jquery-1.6.2.js" type="text/javascript"></script>
```

4.3.2 jQuery 选择器

在 jQuery 中，无论使用哪种类型的选择器，都需要从一个美元符号和一对圆括号开始：$（）。所有能在样式表中使用的选择器，都能放到这个圆括号中的引号内。$（）是最经常使用的一个函数，其实，$（）就是 jQuery 的别名，只不过$（）语法更简洁，语句更短。

jQuery 使用类似于 CSS 的选择器查找页面上的元素，然后进行各种操作，如更改样式、读取或设置文本、隐藏显示等，jQuery 支持多种选择器，常用有以下几种：

1. 元素选择器

元素选择器能够选择某一元素作为参数，如 a、img、input 等。例如，tr 表示选择所有 tr 元素，div 表示选择所有 div 元素。例如给所有表格加上高亮的效果，即随着鼠标移动始终突出显示鼠标下面的行，则可以使用以下代码实现：

```
$("tr").hover(
     function(){$(this).addClass("hightlight");},
     function(){$(this).removeClass("hightlight");}
);
```

上述代码中$("tr")就选择了页面中所有的 tr 元素，然后为其鼠标进入和移出事件分别编写事件处理程序，添加和移除高亮显示样式。

2. ID 选择器

ID 选择器是指根据一个元素的 ID 来选择元素。ID 选择器的语法是以#开头，紧接着是元素的 ID。例如$("#button1")将选择 ID 为 button1 的元素。ID 选择器有时也和元素选择器一起使用，例如$("div #main")选中 ID 为 main 的 div 元素。

3. 类选择器

类选择器可以根据元素的 class 属性进行选择。类选择器的语法是以.开头，紧接着是 CSS 类名。例如$(".important")将选择页面上所有应用了 important 类的元素。

4．后代选择器

如果在一个选择器后面有一个空格，再跟另外一个选择器，则表示选择包含在第一个选择器中的第二个选择器。例如，$("div p")将选择 div 中出现的所有 p 元素。

5．子元素选择器

如果一个选择器后面是一个大于号>，后面再跟另一个选择器，则表示选择直接包在第一个选择器中的第二个选择器。注意，这种选择器与后代选择器不同，后代选择器只要求后代关系，而不是直接或间接关系，而子元素选择器只选择直接元素。例如：

```
$(document).ready(function(){
    $('#selected-plays > li').addClass('horizontal');
});
```

上述位于$（）函数中的选择器的含义是，查找 ID 为 selected-plays 的元素（#selected-plays）的子元素（>）中所有的列表项（li）。

4.3.3　jQuery 中关于 DOM 的操作

使用 jQuery 最常见的功能就是对 DOM 元素的操作，包括读取和设置其文本，修改其样式，添加、删除子元素等。本节将介绍如何通过 jQuery 实现常见的 DOM 操作。

对于 DOM 元素的一个常见操作就是读取或设置其内容，例如，获取 TextBox 的值，设置 div 标记中的内容，获取或设置表格某单元格内文本等。jQuery 主要提供两个函数（html()和 val()）来实现这些功能。

第一个函数是 html()函数，该函数有两个作用，如果函数没有参数，则获取元素内的

html 内容：如果函数有一个参数，则将元素内容设置为字符串参数所表示的 HTML 元素。其语法如下：

```
html()                //返回元素内部的 html 内容
html(content)         //content 为字符串，将元素内容设置为 content 参数所指定值
```

【例 4.55】使用 html 元素处理元素内容。

本例演示 html()函数读取和设置元素内容。页面上有两个 div 和两个按钮，单击第 1 个按钮可以实现读取第 1 个 div 内容并以消息框的形式显示出来。单击第 2 个按钮可以将第 1 个 div 的内容复制到第 2 个 div 中。代码如下：

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>jQuery 函数示例</title>
   <style type = "text/css" >
      div#main > div
     {
        border :1px dashed silver;
        margin :10px;
        height :100px;
     }
   </style>
  <script src="js/jquery-1.4.2.min.js" type="text/javascript"></script>
    <script type = "text/javascript" >
      $(function () {
         $("#showButton").click(function () {
            var content = $("#div1").html();
               alert(content);
         });
          $("#copyButton").click(function () {
             var content = $("#div1").html();
               $("#div2").html(content);
          });
      });
     </script>
</head>
<body>
<div id = "main">
  <input type = "button" value = "显示第一个div内容" id = "showButton" /><br />
  <input type="button" value="复制第一个 div 内容到第二个 div 中" id="copyButton"
/><br />
<div id = "div1">
  <p>这是第一个div里的文字</p>
</div>
<div id = "div2">
   原来的内容
```

```
</div>
</div>
</body>
</html>
```

页面运行结果如图 4-19 所示:

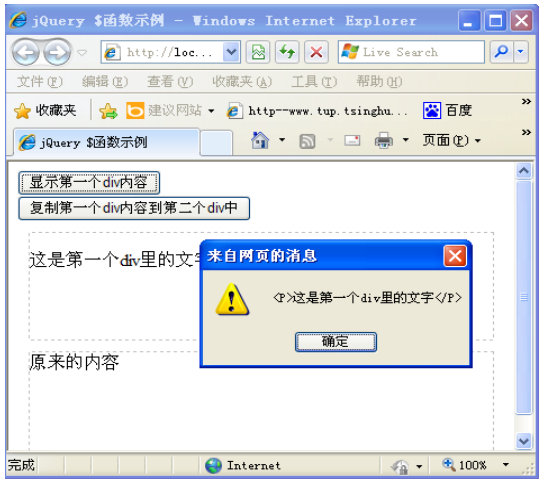

图 4-19　使用 html()函数处理元素内容

jQuery 第 2 个常用的操作元素内容的函数为 val()函数。这个函数也有两种用法，如果函数没有参数，则获取元素的值；如果函数有参数，则将元素的值设置为参数的值。val()函数通常用来获取和设置 text、button 等 input 元素的值。例如，假设用户在一个 ID 为 yourName 的文本框中输入姓名，则可以使用以下语句向用户问候。

```
var name = $("#yourName").val();
alert("hello"+ name);
```

jQuery 提供了多种途径更改元素的样式。其中一种最简单、最直观的方式是使用 css()函数。css()函数可以设置或获取元素的 css 属性，语法如下：

```
css(name)                          //获取名称为 name 的 css 属性
css(name,value)                    //将名为 name 的 css 属性设置为 value
```

下面的代码是使用 css()函数的例子：

```
var back = $("#div1").css("background-color");        //获取div1的背景色
$("#div2").css("background-color", "#eeffee");       //设置 div2 背景色为浅绿色
```

使用 jQuery 设置元素样式的第 2 种方法是使用 addClass()函数和 removeClass()函数。addClass()向元素添加 CSS 类，removeClass()函数从元素删除 CSS 类。所添加的 CSS 类应该在页面中定义。下面的例子演示 addClass()和 removeClass()函数的使用。

```
$("#div1").addClass("green");                    //向 div1 添加 green 类
$("#div1").removeClass("red");                   //向 div1 删除 red 类
```

与 addClass 和 removeClass 密切相关的还有一个 toggleClass 类，其作用为在添加和移除 CSS 类之间进行切换。如果元素已经应用了 CSS 类，则将其移除，否则添加。其用法如下：

```
$("#div2").toggleClass("test");
```

现在网页上经常见到一种表格的光棒效果，即鼠标移动到表格某行时，该行改变背景色高亮显示，鼠标移出时恢复成普通颜色，从而实现鼠标移入行的始终高亮显示。jQuery 提供了一个很方便的函数来实现这个功能，这个函数就是 hover()函数。hover()函数语法如下：

```
hover(
  function(){/*这里写鼠标进入时要执行的代码*/},
  function(){/*这里写鼠标移出时要执行的代码*/}
)
```

hover()函数有两个参数，这两个参数都是函数，当鼠标进入时执行第一个函数，鼠标移出时执行第二个函数。利用 hover()函数结合 addClass()函数，可以方便的实现鼠标的光棒效果。

【例 4.56】表格效果演示。

本例演示常用的表格效果。首先实现了光棒效果，随着鼠标移动，鼠标所在行总会高亮显示。另外，当单击某行时，还可以在选中和非选中之间进行切换。

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>表格效果</title>
    <style type = "text/css" >
        .selected { background-color :#ddeeff; }
        .highlight{  background-color :#ffffee;}
    </style>
    <script src="js/jquery-1.4.2.min.js" type="text/javascript"></script>
    <script type = "text/javascript" >
        $(function () {
            $("tr").click(function () {
                $(this).toggleClass("selected");
            });
            $("tr").hover(
                function () { $(this).addClass("highlight"); },
                function () { $(this).removeClass("highlight"); }
            );
        });
    </script>
</head>
<body>
```

```
<div id = "main">
    <table >
        <tr>
            <th>产品名称</th> <th>产品价格</th> <th>产品生产日期</th>
        </tr>
        <tr>
            <td>MP3</td> <td>500</td> <td>2007.1.25</td>
        </tr>
         <tr>
            <td>笔记本电脑</td> <td>5000</td> <td>2007.5.25</td>
        </tr>
        <tr>
 <td>iphone4</td> <td>4999</td> <td>2010.6.25</td>
        </tr>
    </table>
</div>
</body>
</html>
```

运行结果如图 4-20 所示。

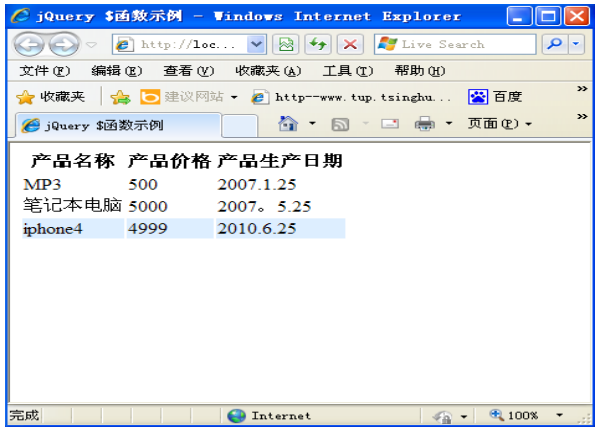

图 4-20　表格演示效果

关于 jQuery 中对 DOM 操作的方法还有很多，表 4-13 列出了一些常用的访问 DOM 元素的方法：

表 4-13 jQuery 访问 DOM 常见方法

函数	描述
.get()	获得由选择器指定的 DOM 元素
.index()	返回指定元素相对于其他指定元素的 index 位置
.size()	返回被 jQuery 选择器匹配的元素的数量
.toArray()	以数组的形式返回 jQuery 选择器匹配的元素

.attr()	获得匹配元素的第一个元素指定的属性
.addClass()	增加类名，即是增加 class 属性
.removeClass()	删除类名，即是删除 class 属性
.toggleClass()	切换类名（存在则删除，不存在则增加）

4.3.4 jQuery 事件

JavaScript 和 HTML 之间的交互是通过用户和浏览器操作页面时引发的事件来处理的。当文档或者它的某些元素发生某些变化或操作时，浏览器会自动生成一个事件。例如当浏览器装载完一个文档后，会生成事件；当用户单击某个按钮时，也会生成事件。虽然利用传统的 JavaScript 事件能完成这些交互，但 jQuery 增加并扩展了基本的事件处理机制。jQuery 不仅提供了更加优雅的事件处理语法，而且极大地增强了事件处理能力。

以浏览器装载文档为例，在页面加载完毕后，浏览器会通过 JavaScript 为 DOM 元素添加事件。在常规的 JavaScript 代码中，通常使用 window.onload 方法，而在 jQuery 中，使用的是$(document).ready()方法。$(document).ready()方法是事件模块中最重要的一个函数，可以极大地提高 Web 应用程序的响应速度。jQuery 就是用$(document).ready()方法来代替传统 JavaScript 的 window.onload 方法的。

1. 事件的绑定

在文件装载完成后，如果打算为元素绑定事件来完成某些操作，则可以使用 bind()方法来配对元素进行特定事件的绑定，bind()方法的调用格式为：

```
bind（type [, data], fn）;
```

Bind()方法有 3 个参数，第一个参数是事件类型，包括：blur、focus、load、resize、scroll、unload、click、mousedown、mouseup、mousemove、mouseover、mouseout、mouseenter、mouseleave、change、select、submit、keydown、keypress、keyup 和 error 等，当然也可以是自定义名称。第二个参数为可选参数，作为 event.data 属性传递给事件对象的额外数据对象。第三个参数则是用来绑定的处理函数。

下面通过一个例子来了解 bind()方法的用法。

```
$(document).ready(function(){
    $('#panel').bind('click',function(){ //绑定 click 事件
       $('body').addClass('large');      //单击 ID 为#panel 的元素时添加 large 类样式
    });
});
```

2. 合成事件

jQuery 有两个合成事件——hover()方法和 toggle()方法，都属于 jQuery 自定义的方法。hover()方法的语法结构为：

```
hover(enter, leave);
```

Hover()方法用于模拟光标悬停事件。当光标移动到元素上时，会触发指定的第 1 个函

数(enter)；当光标移出这个元素时，会触发指定的第 2 个函数(leave)。例如：

```
$(document).ready(function(){
    $('#panel').hover(function(){
      $(this).addClass('large');        //鼠标移动到该元素上时添加 large 类样式
   }, function(){
      $(this).removeClass('large');     //鼠标移出该元素时移除 large 类样式
    });
});
```

toggle()方法的语法结构为:

```
toggle(fn1,fn2,…fnN);
```

toggle()方法用于模拟鼠标连续单击事件.第 1 次单击元素，触发指定的第 1 个函数(fn1)；当再次单击同一元素时，则触发指定的第 2 个函数(fn2)；如果有更多函数,则依次触发，直到最后一个。随后的每次单击重复对这几个函数的轮番调用。

```
$(document).ready(function(){
    $('#panel').toggle(function(){
      $(this).addClass('large');
   }, function(){
      $(this).removeClass('large');     //鼠标连续单击时添加和移除 large 类样式
    });
});
```

jQuery 提供了许多创建动态效果的方法，下例的 toggle（）方法与其他方法一起轻松地实现了动画效果。

【例 4.57】jQuery 动画效果演示。本例演示一个简单的 jQuery 动画效果。动画折叠展开一个层，点击后淡出，然后 DIV 又发生形状的变化。

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>jquery先淡出再变形的动画</title>
<style type="text/css">
*{margin: 0;padding: 0;}
body { font-size: 13px; line-height: 130%; padding: 60px }
#panel { width: 300px; border: 1px solid #0050D0 }
.head { padding: 5px; background: #96E555; cursor: pointer }
.content { padding: 10px; text-indent: 2em; border-top: 1px solid
#0050D0;display:block; }
</style>
<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
<script type="text/javascript">
$(function(){
    $("#panel h5.head").toggle(function(){
        $(this).next("div.content").fadeOut();
    },function(){
        $(this).next("div.content").fadeIn();
    })
})
</script>
</head>
<body>
<div id="panel">
    <h5 class="head">简单的jQuery动画</h5>
    <div class="content">
        展示使用jQuery生成动画效果的一个小例子，让一个Div层先淡出然后再发生形变，最后
折叠消失。</div>
</div>
</body>
</html>
```

3. jQuery 名称冲突

在某些情况下，可能有必要在同一个页面中使用多个 JavaScript 开发库。由于很多库都使用$标识符，因此就需要某种方式来避免名称冲突。

为解决这个问题，jQuery 提供了一个名叫.noConflict()方法，调用该方法可以把对$标识符的控制权交还给其他库。使用.noConflict()方法的一般模式如下：

```
<script src="prototype.js" type="text/javascript"></script>
<script src="jquery.js" type="text/javascript"></script>
<script type="text/javascript">
  jQuery.noConflict();
</script>
<script src="myscript.js" type="text/javascript"></script>
```

首先，包含 jQuery 之外的库（这里是 Prototype 库）。然后，包含 jQuery 库，取得对$的使用权。接着，调用.noConflict()方法让出$，以便将控制权交给最先包含的库（Prototype）。这样，就可以在自定义脚本中使用两个库了。但是，在需要使用 jQuery 方法时，必须记住要用 jQuery 而不是$来调用。

由于 jQuery 是为处理 HTML 事件而特别设计的，所以为了让代码更恰当且更易维护，最好遵循以下原则：

- 把所有 jQuery 代码置于事件处理函数中
- 把所有事件处理函数置于文档就绪事件处理器中
- 把 jQuery 代码置于单独的.js 文件中
- 如果存在名称冲突，则重命名 jQuery 库

表 4-14 列出了 jQuery 常用事件方法：

表 4-14 jQuery 常用事件方法

方法	描述
bind()	向匹配元素添加一个或更多事件处理器
blur()	触发、或将函数绑定到指定元素的 blur 事件
change()	触发、或将函数绑定到指定元素的 change 事件
click()	触发、或将函数绑定到指定元素的 click 事件
dblclick()	触发、或将函数绑定到指定元素的 double click 事件
delegate()	向匹配元素的当前或未来的子元素附加一个或多个事件处理器
error()	触发、或将函数绑定到指定元素的 error 事件
focus()	触发、或将函数绑定到指定元素的 focus 事件
keydown()	触发、或将函数绑定到指定元素的 key down 事件
keypress()	触发、或将函数绑定到指定元素的 key press 事件
keyup()	触发、或将函数绑定到指定元素的 key up 事件
load()	触发、或将函数绑定到指定元素的 load 事件
ready()	文档就绪事件（当 HTML 文档就绪可用时）
resize()	触发、或将函数绑定到指定元素的 resize 事件
scroll()	触发、或将函数绑定到指定元素的 scroll 事件
submit()	触发、或将函数绑定到指定元素的 submit 事件
toggle()	绑定两个或多个事件处理器函数，当发生轮流的 click 事件时执行
trigger()	所有匹配元素的指定事件

4.4　DHTML 综合编程实践

4.4.1　广告条定时滚动

功能描述：定时滚动广告条。共有 3 个广告条，每个广告条中有 6 个超链接。页面显示 10s 后才开始广告滚动。将鼠标放到广告条上就停止滚动，鼠标离开广告条就滚动。

```
<HTML><HEAD><TITLE></TITLE>
<META http-equiv=Content-Type content="text/html; charset=gb2312">
<STYLE type="text/css">
TD {FONT-SIZE: 12px; COLOR: #000000; FONT-FAMILY: 宋体}
a{ color: #000000;}
A:link {TEXT-DECORATION: none}
A:visited {TEXT-DECORATION: none}
A:active {TEXT-DECORATION: none}
A:hover {TEXT-DECORATION: underline}
</STYLE>
<META content="MSHTML 6.00.2800.1515" name=GENERATOR></HEAD>
<BODY leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<TABLE id="table1" style="BORDER-COLLAPSE: collapse" cellPadding=1 width=468
border=0>
  <TBODY>
  <TR>
    <TD width="380">
     <DIV id="so1">
     <table   width="380"   border="0"   align="center"   cellpadding="0"
cellspacing="0">
       <TBODY> <tr>
       <td height="20"><a href="url1" target="_blank">重庆大学</a></td>
       <td height="20"><a href="url2" target="_blank">精美图片</a></td>
       <td height="20"><a href="url3" target=_blank>学习论坛</a></td>
       <td height="20"><a href="url4" target=_blank>电视娱乐</a></td>
       <td height=20><a href="url5" target=_blank>娱乐星闻</a></td>
```

```
       <td height=20><a href="url6" target=_blank>happyb 中文</a></td>
       </tr> </TBODY>
       </table>

     <TABLE width=380 cellSpacing=0 cellPadding=0 border=0>
       <TBODY><tr>
          <td height=20><a href="url1" target=_blank>小木屋</a></td>
         <td height=20><a href="url2" target=_blank>易物在线超</a></td>
         <td height=20><a href="url3" target=_blank>通州论坛</a></td>
         <td height=20><a href="ur14" target=_blank>QQ 宝典资源</a></td>
         <td height=20><a href="url5" target=_blank>冰霄娱乐</a></td>
         <td height=20><a href="url6" target=_blank>时尚元素网</a></td>
          </tr></TBODY>
       </table>

     <TABLE width=380 cellSpacing=0 cellPadding=0 border=0>
       <TBODY><tr>
         <td height=20><a href="url1" target=_blank>动感 FLASH</a></td>
```

```
        <td height=20><a href="url2" target=_blank>在线 MTV 欣</a></td>
        <td height=20><a href="url3" target=_blank> MP3 音乐</a></td>
        <td height=20><a href="url4" target=_blank>心动游戏网</a></td>
        <td height=20><a href="url5" target=_blank>07007 音乐</a></td>
        <td height=20><a href="url6" target=_blank>图片广场</a></td>
        </tr></TBODY>
        </TABLE>
     </DIV>
<DIV id=so2 style="Z-INDEX:1;VISIBILITY:hidden;POSITION:absolute">
</DIV></TD></TR></TBODY></TABLE>
<SCRIPT type="text/javascript">
marqueesHeight=20;
stopscroll=false;
document.all.so1.scrollTop=0;
with(so1){
  style.width=380;
  style.height=marqueesHeight;
  style.overflowX='visible';
  style.overflowY='hidden';
  noWrap=true;
  onmouseover=new Function('stopscroll=true');
  onmouseout=new Function('stopscroll=false');
}

preTop=0; currentTop=0; stoptime=0; //参数初始化

function init_srolltext(){
document.all.so2.innerHTML='';
```

```
document.all.so2.innerHTML+=document.all.so1.innerHTML;
document.all.so1.innerHTML=document.all.so2.innerHTML+document.all.so2.
innerHTML;
setInterval('scrollUp()',1); //设置每秒钟就更换一次广告
}

function scrollUp(){
  if(stopscroll==true) return;
  currentTop+=1;
  if(currentTop==21) {
  stoptime+=1;
  currentTop-=1;
  if(stoptime==150) {
  currentTop=0;
  stoptime=0;  }
}
else {
```

```
  preTop=document.all.so1.scrollTop;
  document.all.so1.scrollTop+=1;
  if(preTop==document.all.so1.scrollTop){
  document.all.so1.scrollTop=document.all.so2.offsetHeight-marqueesHeight;
  document.all.so1.scrollTop+=1;}
}
}
setTimeout('init_srolltext()', 10000) //页面显示秒钟后才开始广告条滚动
</script>
</BODY></HTML>
```

4.4.2 通过 URL 传递参数

功能描述：有两个 HTML 页面文档，分别是 start.htm 和 index.htm，要从 start.htm 中将相关参数通过 url 地址传送到 index.htm 页面，在 index.htm 中显示传过来的参数。在 url 地址中用问号后附带参数的方法传递参数，每个参数之间用&相隔。

例如，"url=http://aa.com/index.htm?email=pp@cqu.edu.cn&name=wcl"中传递了两个参数 email 和 name，它们的值分别是 pp@cqu.edu.cn 和 wcl。

start.htm 文件内容如下：

```
<html><head>
<title>通过 URL 传递参数</title>
</head>
<body>
<script type="text/javascript">
window.open("index.htm?email=pp@cqu.edu.cn&name=wcl","_self");
</script>
```

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

index.htm 文件内容如下：

```
<!-- 下面给出了一个带有详细注释的具体示例源代码。注意：querystring 是一个实用函数，可
以在网页中直接引用，然后在网页中使用 Request["名称"] 即可获取用户输入的有关信息内容。
-->
<Html><Head>
<Meta http-equiv="Content-Type"content="text/html; charset=gb2312" >
<Title>示例</Title>
<script type="text/javascript">
function QueryString()
{//构造参数对象并初始化
 var name,value,i;
 var str=location.href;                    //获得浏览器地址栏 URL 串
 var num=str.indexOf("?")
 str=str.substr(num+1);                    //截取“?”后面的参数串
 var arrtmp=str.split("&");                //将各参数分离形成参数数组
```

```
 for(i=0;i < arrtmp.length;i++){
   num=arrtmp[i].indexOf("=");
   if(num>0){
    name=arrtmp[i].substring(0,num);    //取得参数名称
    value=arrtmp[i].substr(num+1);      //取得参数值
    this[name]=value;                   //定义对象属性并初始化
   }
 }
}
var Request=new QueryString();          //使用 new 运算符创建参数对象实例
</script>
</Head>
<Body>
<script>
var newElement=document.createElement("div");      //创建 div 对象
var str="<u>"+Request["name"]+"</u>,欢迎光临!<br>您的 E-mail 是:<u>"+Request
["email"]+"</u>";          //利用 Request["字段名称"]获取参数内容
newElement.innerHTML=str;
document.body.appendChild(newElement);             //向文档添加 div 对象
</script>
</Body>
</Html>
```

4.4.3　超文本编辑器及其与 Word 的互操作

功能描述：用 Iframe 帧标记实现文本和超文本编辑器功能，可以将编辑框中的内容自动放到 Word 中让用户编辑；可将用户在 Word 中编辑过的文档内容自动放到编辑框中。

```
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>具有超文本编辑功能的文本编辑框</title></head>
<script language="javascript" >
var WordApp;
var wordflag=0;
var myRange;
function OpenWord() //在 Word 中编辑
{
  onerror=errormsg;
  WordApp = new ActiveXObject("Word.Application");
  WordApp.Application.Visible = true;
  var mydoc=WordApp.Documents.Add("",0,1);
  myRange =mydoc.Range(0,1)
  if(myiframe.document.body.innerText=="")
  {
```

```
    myiframe.document.body.innerText=" ";
  }
  var sel=myiframe.document.body.createTextRange();
  sel.select();
  myiframe.document.execCommand('Copy');
  sel.moveEnd('character');
  myRange.Paste();
  WordApp.ActiveWindow.ActivePane.View.Type=3;     // Word 视图模式为页面
  WordApp.Selection.EndKey(5);
  WordApp.Selection.InsertBreak(7);                // 插入分页符
  WordApp.Selection.HomeKey (5);
  WordApp.Selection.TypeText("重庆市房产面积测算报告书");

}
function LoadWord() //更新编辑文档
{
 onerror=errormsgl;
 WordApp.Selection.WholeStory()
 WordApp.Selection.Copy()
 WordApp.Selection.Delete()
 myRange.Paste()
 newDocument()
 myiframe.document.execCommand('Paste')
 for(i=WordApp.Documents.Count;i>0;i--){
   WordApp.Documents(i).Close(0);
 }
 WordApp.Application.quit()
}
```

```
function errormsg()
{
  alert("请确认是否装有 Word 如果装有请确认\r 是否 IE 的安全选项\"本地 Intranet\"级别
\r 已被设为\"低\"否则将不能使用此功能,步骤如下:\r 工具-\>Internet 选项->安全\r-\>本
地 Intranet-\>自定义级别-\>第二项选为启用或提示");
  return true;
}
function errormsgl()
{
 alert("Word 没打开或者 Word 已经关闭!请打开 Word 编辑后再更新!")
 return true;
}
function initpage()            //初始化正文编辑器
 {
 if (document.all.myiframe) {
 myiframe.document.designMode="On";
```

```
 myiframe.document.open();
 myiframe.document.write("<div><br></div>");
 myiframe.document.close();
 }
}
function newDocument()          //清除正文内容
 {
 myiframe.document.designMode="On";
 myiframe.document.open();
 myiframe.document.write("");
 myiframe.document.close();
 myiframe.focus()
 }

 var format="HTML";
function swapModes()            //切换编辑状态
 {
    if (format=="HTML") {
        myiframe.document.body.innerText = myiframe.document.body.innerHTML
        myiframe.document.body.style.fontFamily = "arial";
        myiframe.document.body.style.fontSize = "10pt";
        format="Text";
    }
    else {
        myiframe.document.body.innerHTML=myiframe.document.body.innerText;
        myiframe.document.body.style.fontFamily = "";
        myiframe.document.body.style.fontSize ="";
        format="HTML";
```

```
    }
}
</script>
<body onload="initpage();" bgcolor="#FFFFCC" >
<input id="B0" type="button" value="切换编辑状态" onclick="swapModes();">
<input id="B1" type="button" value="打开 Word 编辑" onclick="OpenWord();">
<input id="B2" type="button" value="从 Word 中调入" onclick="LoadWord();">
<input id="B3" type="button" value="清空" onclick="newDocument();"><br />
<IFRAME id="myiframe" marginWidth=2 marginHeight=2 width="50%" height="50%"
></IFRAME>
</body></html>
```

4.4.4　表格的美化

功能描述：美化表格的 JavaScript 自定义函数。其功能包括：①可设置表格行的交替

颜色；②当鼠标在表格上移动时，以某颜色区别显示鼠标指针所在表格行；③当鼠标选中某行时，高亮显示该行。

```
// 标题: beautify tables
// 描述: 美化表格。1.设置表格行的交替颜色; 2.当鼠标在表格上移动时, 以某颜色区别显示鼠标
//指针所在表格行; 3.当鼠标选中某行时, 高亮显示该行。
// 版本: 1.0
// 日期: 2002-03-04

function beau_tables (
str_tableid,                // table 的 id 属性(必选项)
num_header_offset,          // 表格头部无需美化的行数(可选项)
num_footer_offset,          //表格尾部无需美化的行数(可选项)
str_odd_color,              // 奇数行背景颜色(可选项)
str_even_color,             // 偶数行背景颜色(可选项)
str_mover_color,  // background color for rows with mouse over (opt.)
str_onclick_color // background color for marked rows (opt.)
) {
 // 浏览器不支持 DOM 则返回
if (typeof(document.all) != 'object') return;
// 验证传递的参数
   if (!str_tableid) return alert ("没有指定表格的 ID 参数! ");
   var obj_tables = (document.all ? document.all[str_tableid] :
document.getElementById(str_tableid));
//如果有多个表格, 则可能返回是数组
   if (!obj_tables) return alert ("找不到 ID 为"+str_tableid+"的表格");

   // 为可选输入项指定默认参数
   var col_config = [];
```

```
   col_config.header_offset = (num_header_offset ? num_header_offset : 0);
   col_config.footer_offset = (num_footer_offset ? num_footer_offset : 0);
   col_config.odd_color = (str_odd_color ? str_odd_color : '#ffffff');
   col_config.even_color = (str_even_color ? str_even_color : '#dbeaf5');
   col_config.mover_color = (str_mover_color ? str_mover_color : '#6699cc');
   col_config.onclick_color = (str_onclick_color ? str_onclick_color :
'#4C7DAB');

   // init multiple tables with same ID
   if (obj_tables.length)
   for (var i = 0; i < obj_tables.length; i++)     //对多个表格依次处理
   tt_init_table(obj_tables[i], col_config);
   else
   tt_init_table(obj_tables, col_config);          //对单个表格的处理
}
```

```
function tt_init_table (obj_table, col_config)
{
   var col_lconfig = [],
   col_trs = obj_table.rows;
      //为表格每行设置相关属性和事件，跳过表头和表尾不作处理的行
   for (var i=col_config.header_offset; i<col_trs.length - col_config.footer_
offset;i++) {
   col_trs[i].config = col_config;          //为 TR 追加设置的自定义属性 当前的行
   col_trs[i].lconfig = col_lconfig;        //为 TR 追加设置的自定义属性 移到的行
   col_trs[i].set_color = tt_set_color;     //为 TR 追加设置的自定义事件

   col_trs[i].onmouseover = tt_mover;
   col_trs[i].onmouseout = tt_mout;
   col_trs[i].onmousedown = tt_onclick;
   //为 TR 追加设置的自定义属性
   col_trs[i].order = (i - col_config.header_offset) % 2;
   col_trs[i].onmouseout();
   }
}

function tt_set_color(str_color) {
   this.style.backgroundColor = str_color;
}

// 事件处理程序
function tt_mover () {
   if (this.lconfig.clicked != this)
       this.set_color(this.config.mover_color);
```

```
}
function tt_mout () {
   if (this.lconfig.clicked != this)
       this.set_color(this.order ? this.config.odd_color : this.config.even_
color);
}
function tt_onclick () {
   if (this.lconfig.clicked == this) {
       this.lconfig.clicked = null;
       this.onmouseover();
   }
   else {
       var last_clicked = this.lconfig.clicked;
       this.lconfig.clicked = this;
       if (last_clicked) last_clicked.onmouseout();
```

```
        this.set_color(this.config.onclick_color);
    }
}
```

思考练习题

1．DHTML 的组成是什么？
2．JavaScript 脚本语言有哪些特点？与 Java 语言的区别是什么？
3．JavaScript 中如何创建对象？
4．JavaScript 主要内置对象有哪些？如何利用 JavaScript 进行事件编程？
5．如何通过 HTML DOM 操纵 HTML 元素？
6．HTML DOM 树在 Web 开发中有什么作用？
7．jQuery 中子元素选择器与后代选择器有什么不同？
8．怎样通过 jQuery 操作 DOM 元素？
9．如何避免 jQuery 的名称冲突？
10．对本章每个例子进行上机练习。