第 7 章　Window 及相关顶级对象

在“JavaScript 基于对象编程”一章中，读者基本理解了 Window 相关顶级对象及它们之间的关系。在“文档结构模型(DOM)”一章中，继续加深了这种认识并从对象模型层次关系的角度重点分析了对象的产生过程。本章将从实际应用的角度出发，讨论 Window、Frames、Navigator、Screen、History、Location、Document 等相关顶级对象的属性、语法及如何创建、使用等问题。通过本章的学习，读者应能使用 JavaScript 脚本生成并管理浏览器基本框架，并且熟悉框架之间的通信方法。

7.1 顶级对象模型参考

在 DOM 架构中，Window、Frames、Navigator 等顶级对象产生于浏览器载入文档至关闭文档期间的不同阶段，并起着互不相同且不可代替的作用，如 Window 对象在启动浏览器载入文档的同时生成，与当前浏览器窗口相关，包含窗口的最小最大化、尺寸大小等属性，同时具有关闭窗口、创建新窗口等方法；而 Location 对象以 URL 的形式载入当前窗口，并保存正在浏览的文档的位置及其构成信息，如协议、主机名、端口、路径、URL 的查询字符串部分等，顶级模型的结构如图 7.1 所示。

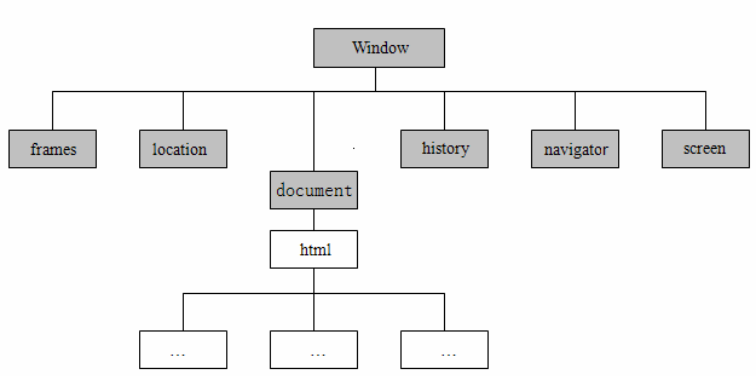

图 7.1 顶级对象模型层次结构

可见，Window 对象在层次中的最上层，而 Document 对象处于顶级对象的最底层。一般说来，Frames 对象在 Window 对象的下层，但当目前文档包含框架集时，该框架集中的每个框架都包含单独的 Window 对象；每个 Window 对象都直接包含一个（或者间接包含多个）Document 对象。首先来了解 Window 对象。

7.2 Window 对象

简而言之，Window 对象为浏览器窗口对象，为文档提供一个显示的容器。当浏览器载入目标文档时，打开浏览器窗口的同时，创建 Window 对象的实例，Web 应用程序开发者可通过 JavaScript 脚本引用该实例，从而进行诸如获取窗口信息、设置浏览器窗口状态或者新

建浏览器窗口等操作。同时，Window 对象提供一些方法产生图形用户界面中用于客户与页面进行交互的对话框（模式或者非模式），并能通过脚本获取其返回值然后决定浏览器后续行为。

由于 Window 对象是顶级对象模型中的最高级对象，对当前浏览器的属性和方法，以及当前文档中的任何元素的操作都默认以 Window 对象为起始点，并按照对象的继承顺序进行访问和相关操作，所以在访问这些目标时，可将引用 Window 对象的代码省略掉，如在需要给客户以警告信息的场合调用 Window 对象的 alert()方法产生警告框，直接使用 alert(targetStr)语句，而不需要使用 window.alert(targetStr)的方法。但在框架集或者父子窗口通信时，须明确指明要发送消息的窗口名称。

7.2.1 交互式对话框

使用 Window 对象产生用于客户与页面交互的对话框主要有三种：警告框、确认框和提示框等，这三种对话框使用 Window 对象的不同方法产生，功能和应用场合也不大相同。

1. 警告框

警告框使用 Window 对象的 alert()方法产生，用于将浏览器或文档的警告信息（也可能不是恶意的警告）传递给客户。该方法产生一个带有短字符串消息和“确定”按钮的模式对话框，且单击“确定”按钮后对话框不返回任何结果给父窗口。此方法的语法如下：

```
window.alert(Str);
alert(Str);
```

其中参数可以是已定义变量、文本字符串或者是表达式等。当参数传入时，将参数的类型强制转换为字符串然后输出：

```
var MyName="YSQ";
var iNum=1+1;
alert("\nHello " +MyName+ ":\n MyResult: 1+1=" +iNum+ "\n");
```

上述代码运行后，弹出警告框如图 7.2 所示。

图 7.2 警告框实例

> 注意：模式对话框由父窗口创建，并使父窗口无效直到该对话框被关闭之后父窗口才能被用户激活，其内部拥有消息循环；非模式对话框由父窗口创建，对话框创建后立即返回，并与父窗口共用一个消息循环，在对话框被关闭之前，父窗口能被激活进行各种操作。

2. 确认框

确认框使用 Window 对象的 conform()方法产生，用于将浏览器或文档的信息（如表单提交前的确认等）传递给客户。该方法产生一个带有短字符串消息和“确定”、“取消”按钮的模式对话框，提示客户选择单击其中一个按钮表示同意该字符串消息与否，“确定”按钮表示同意，“取消”按钮表示不同意，并将客户的单击结果返回。此方法的语法如下：

```
answer=window.confirm(Str);
answer=confirm(Str);
```

其中参数可以是已定义变量、文本字符串或者是表达式等。当参数传入时，将参数的类型强制转换为字符串作为需要确认的问题输出。该方法返回一个布尔值表示消息确认的结果，true 表示客户同意了该消息，而 false 表示客户不同意该消息或确认框被客户直接关闭。考察下面代码：

```
var MyName="YSQ";
var iNum=1+1;
var answer=confirm("\nHello " +MyName+ ":\n MyResult: 1+1=" +iNum+ " ?\n");
if(answer==true)
   alert("\n 客户确认信息 : \n\n"+"          算术运算 1+1=2 成立!");
else
   alert("\n 客户确认信息 : \n\n"+"          算术运算 1+1=2 不成立!");
```

程序运行后，弹出确认框如图 7.3 所示。

图 7.3 确认框实例

若单击“确定”按钮，将弹出警告框如图 7.4 所示。

图 7.4 单击“确定”按钮，弹出警告框

若单击“取消”按钮或直接关闭该确认框，将弹出警告框如图 7.5 所示。

图 7.5 单击“取消”按钮或直接关闭确认框，弹出警告框

值得注意的是，确认框中事先设定的问题的提问方法有可能严重影响这个确认框的实用性，程序员在编制程序时，一定要将问题的返回值代表的客户意图与下一步的动作联系在一起，避免对结果进行完全相反的处理。

3. 提示框

提示框使用 Window 对象的 prompt()方法产生，用于收集客户关于特定问题而反馈的信息，该方法产生一个带有短字符串消息的问题和“确定”、“取消”按钮的模式对话框，提示客户输入上述问题的答案并选择单击其中一个按钮表示确定还是取消该提示框。如果客户单击了“确定”按钮则将该答案返回，若单击了“取消”按钮或者直接关闭则返回 null 值。此方法的语法如下：

```
returnStr=window.prompt(targetQuestion,defaultString);
returnStr=prompt(targetquestion,default string);
```

该方法通过参数 targetQuestion 传入一个字符串，代表需要客户回答的问题。通过参数 defaultString 传入一个默认的字符串，该参数一般可设定为空。当客户填入问题的答案并单击“确定”按钮后，该答案作为 prompt()方法的返回值赋值给 returnStr；当客户单击“取消”按钮时，prompt()方法返回 null。考察如下代码：

```
var answer=prompt("算术运算题目 : 1+1 = ?");
if(answer==2)
   alert("\n 算术运算结果 : \n\n"+"恭喜您,你的答案正确! ");
else if(answer==null)
   alert("\n 算术运算结果 : \n\n"+"对不起,您还没作答! ");
else
   alert("\n 算术运算结果 : \n\n"+"对不起,您的答案错误! ");
```

程序运行后，弹出提示框如图 7.6 所示。

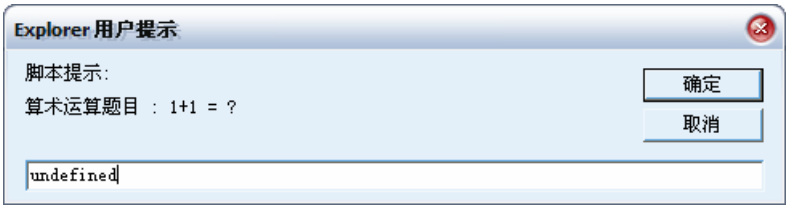

图 7.6 提示框实例

如果在上述提示框填入正确结果“2”，并单击“确定”按钮，弹出警告框如图 7.7 所示。

图 7.7 输入正确结果时弹出警告框

如果在上述提示框输入错误的答案，并单击“确定”按钮，弹出警告框如图 7.8 所示。

图 7.8 输入错误结果时弹出警告框

如果在上述提示框中单击“取消”按钮或直接关闭，弹出警告框如图 7.9 所示。

图 7.9 单击“取消”按钮或直接关闭时弹出警告框

使用 prompt()方法生成提示框返回客户的答案时，应注意考察提示框的返回值，然后采取进一步的动作。

4. 实例：学生信息采集系统

综合以上三种客户与浏览器交互的方法，可编制一个学生信息采集系统，该系统实现学生信息录入功能，并在数据合法性的检验方面进行了充分的考虑。考察系统中采集学生信息并验证数据合法性的页面代码：

```
//源程序 7.1
<!DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page!</title>
<script language="JavaScript" type="text/javascript">
<!--
//检查姓名长度是否合法
function CheckName(str)
{
   var nbool;
   var strLength=str.length;
   if(strLength>3&&strLength<9)
      nbool=true;
   else
      nbool=false;
   return nbool;
}
//检查性别是否合法
function CheckSex(sex)
{
   var nbool;
   if(sex=="male"||sex=="female")
      nbool=true;
   else
      nbool=false;
   return nbool;
}
//检查学号格式是否正确
```

```
function CheckNumber(num)
{
   var nbool;
   var numLength=num.length;
   var index=num.indexOf("2001");
   if(numLength!=8||index==-1)
      nbool=false;
   else
      nbool=true;
   return nbool;
}
//检查邮箱格式是否正确
function CheckEmail(EmailAddr)
{
   var nbool;
   var strLength=EmailAddr.length;
   var index1=EmailAddr.indexOf("@");
   var index2=EmailAddr.indexOf(".",index1);
   if(index1==-1||index2==-1||index2<=index1+1||index1==0||index2==strLength-1)
      nbool=false;
   else
      nbool=true;
   return nbool;
}
//操作函数
function MyMain( )
{
   var MyName=prompt("姓名 : (4 到 8 个字符)");
   var nName=CheckName(MyName);
   while(nName==false)
   {
      var NameMsg="姓名字段验证 : \n\n";
      NameMsg+="结果 : 格式错误.\n";
      NameMsg+="格式 : 长度必须为 4 到 8 个字符.\n";
      NameMsg+="处理 : 单击“确定”按钮返回修改.\n";
      alert(NameMsg);
      MyName=prompt("姓名 : (4 到 8 个字符)");
      nName=CheckName(MyName);
   }
   var MySex=prompt("性别 : (male 或 female)","male");
   var nSex=CheckSex(MySex);
   while(nSex==false)
   {
      var SexMsg="性别字段验证 : \n\n";
      SexMsg+="结果 : 格式错误.\n";
      SexMsg+="格式 : 必须为"male"或"famale".\n";
      SexMsg+="处理 : 单击“确定”按钮返回修改.\n";
      alert(SexMsg);
      MySex=prompt("性别 : (male 或 female)","male");
      nSex=CheckSex(MySex);
   }
   var MyNum=prompt("学号 : (形如 2001****)","2001");
   var nNum=CheckNumber(MyNum);
```

```
  while(nNum==false)
  {
    var NumMsg="学号字段验证 : \n\n";
    NumMsg+="结果 : 格式错误.\n";
    NumMsg+="格式 : 必须形如"2001****"格式. \n";
    NumMsg+="处理 : 单击“确定”按钮返回修改.\n";
    alert(NumMsg);
    MyNum=prompt("学号 : (形如 2001****)","2001");
    nNum=CheckNumber(MyNum);
  }
  var MyEmail=prompt("邮箱 : (形如 zangpu@gmail.com)","@");
  var nEmail=CheckEmail(MyEmail);
  while(nEmail==false)
  {
    var EmailMsg="邮箱字段验证 : \n\n";
    EmailMsg+="结果 : 格式错误.\n";
    EmailMsg+="格式 :1、邮件地址中同时含有'@'和'.'字符.\n";
    EmailMsg+="          2、'@'后必须有'.'，且中间至少间隔一个字符. \n";
    EmailMsg+="          3、'@'不为第一个字符，'.'不为最后一个字符.\n";
    EmailMsg+="处理 : 单击“确定”按钮返回修改.\n";
    alert(EmailMsg);
    MyEmail=prompt("邮箱 :","@");
    nEmail=CheckEmail(MyEmail);
  }
  var msg="\n 学生信息 : \n\n"
  msg+="姓名 :"+MyName+"\n";
  msg+="性别 :"+MySex+"\n";
  msg+="学号 :"+MyNum+"\n";
  msg+="邮箱 :"+MyEmail+"\n";
  alert(msg);
}
//-->
</script>
</head>
<body>
<form name="MyForm">
  <input type="button" value="测试" onclick="MyMain()">
</body>
</html>
```

程序运行后，单击页面中的“测试”按钮，弹出“姓名”提示框提示学生输入姓名，如图 7.10 所示。

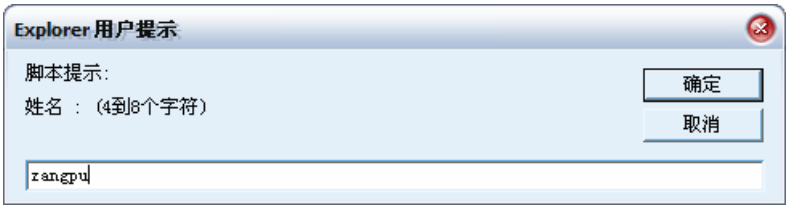

图 7.10 “姓名”提示框

当学生输入正确格式的姓名之后，弹出“性别”提示框提示学生输入性别，如图 7.11 所示。

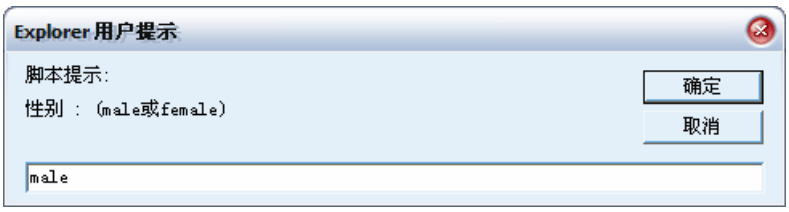

图 7.11 “性别”提示框

当学生输入正确格式的性别之后，弹出“学号”提示框提示学生输入学号，如图 7.12 所示。

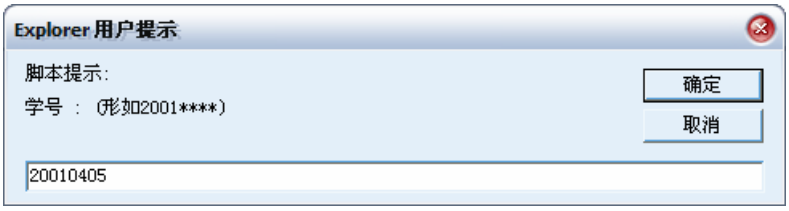

图 7.12 “学号”提示框

当学生输入正确格式的学号之后，弹出“邮箱”提示框提示学生输入邮箱，如图 7.13 所示。

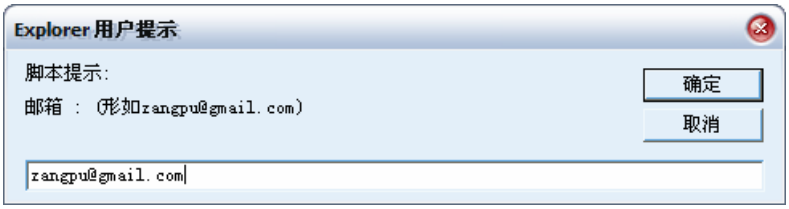

图 7.13 “邮箱”提示框

当学生完成前几项信息输入后，将弹出“学生信息”提示框，该框汇总了前面学生输入的信息，如图 7.14 所示。

图 7.14 “学生信息”提示框

上述的过程演示了如何在页面中采集客户的信息并保存下来，实际应用中，还应该加入判断客户输入信息是否符合要求的代码。本系统使用函数的形式判断，如果符合特定格式，则进入下一个收集信息步骤；若不符合该特定格式，则返回收集当前项信息的提示框继续执行下去。

如果“姓名”字段格式不满足要求，则弹出警告框提示学生输入正确的当前项信息，如图 7.15 所示。

图 7.15 “姓名”字段不正确时，弹出警告框

单击“确定”按钮后，返回“姓名”字段信息输入提示框。上述步骤反复，直到“姓名”字段正确后进入下一步。

如果“性别”字段格式不满足要求，则弹出警告框提示学生输入正确的当前项信息，如图 7.16 所示。

图 7.16 “性别”字段不正确时，弹出警告框

单击 i“确定”按钮后，返回“性别”字段信息输入提示框。上述步骤反复，直到“性别”字段正确后进入下一步。

如果“学号”字段格式不满足要求，则弹出警告框提示学生输入正确的当前项信息，如图 7.17 所示。

图 7.17 “学号”字段不正确时，弹出警告框

单击“确定”按钮后，返回“学号”字段信息输入提示框。上述步骤反复，直到“学号”字段正确后进入下一步。

如果“邮箱”字段格式不满足要求，则弹出警告框提示学生输入正确的当前项信息，如图 7.18 所示。

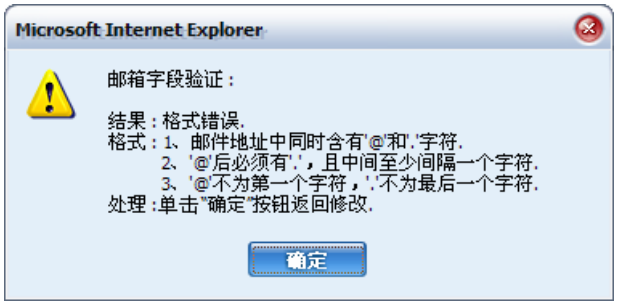

图 7.18 “邮箱”字段不正确时，返回警告框

上面的学生信息采集系统融合了采集、验证等功能，可以将其适当扩展，如将采集的数据提交给目标页面，或者直接录入数据库备用等。

Window 对象提供几种客户与页面交互，并通过对话框采集客户信息的途径。通过综合使用这几种方法，可实现页面的交互性、动态性等要求。

7.2.2 设定时间间隔

Window 对象提供 setInterval()方法用于设定时间间隔，用于按照某个指定的时间间隔去周期触发某个事件，典型的应用如动态状态栏、动态显示当前时间等，该方法的语法如下：

```
TimerID=window.setTimeout(targetProcess,itime);
TimerID=setTimeout(targetProcess,itime);
```

其中参数 targetProcess 指目标事件，参数 itime 指间隔的时间，以毫秒(ms)为单位。设定时间间隔的操作完成后，返回该时间间隔的引用变量 TimerID。

同时，Windows 对象提供 clearInterval()方法用于清除该间隔定时器使目标事件的周期触发失效，该方法语法如下：

```
window.clearInterval(TimerID);
```

该方法接受唯一的参数 TimerID，指明要清除的间隔时间引用变量名。考察如下设定和停止动态状态栏的代码：

```
//源程序 7.2
<!DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page!</title>
<script language="JavaScript" type="text/javascript">
<!--
var TimerID;
var dir=1;
var str_num=0;
//用于动态显示的目标字符串
var str="Welcome To JavaScript World!";
//设定动态显示的状态栏信息
function startStatus()
{
   var str_space="";
   str_num=str_num+1*dir;
```

```
    if(str_num>30 || str_num<0)
    {
       dir=-1*dir;
    }
    for(var i=0;i<str_num;i++)
    {
       str_space+=" ";
    }
    window.status=str_space+str;
}
//状态栏滚动开始
function MyStart()
{
    TimerID=setInterval("startStatus();",100);
}
//状态栏滚动结束，并更新状态栏
function MyStop()
{
    clearInterval(TimerID);
    window.status="The Moving Status have been stopped!";
}
//-->
</script>
</head>
<body onload="window.status='Original Status!'">
<br>
<center>
<p>单击对应的按钮，实现动态状态栏的滚动与停止!</p>
<form name="MyForm">
    <input type="button" value="开始状态栏滚动" onclick="MyStart()"><br>
    <input type="button" value="停止状态栏滚动" onclick="MyStop()"><br>
</form>
</center>
</body>
</html>
```

代码运行后，出现如图 7.19 所示的页面：

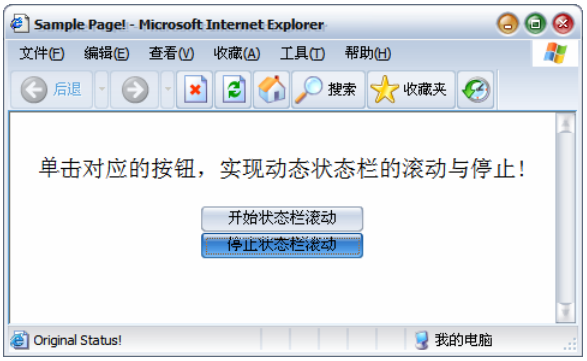

图 7.19 载入页面时出现原始界面

此时的状态栏显示为“Original Status!”，单击“开始状态栏滚动”按钮后，状态栏显示为“Welcome To JavaScript World!”，并按照 setInterval()方法设定的时间间隔左右滚动，如

图 7.20 所示。

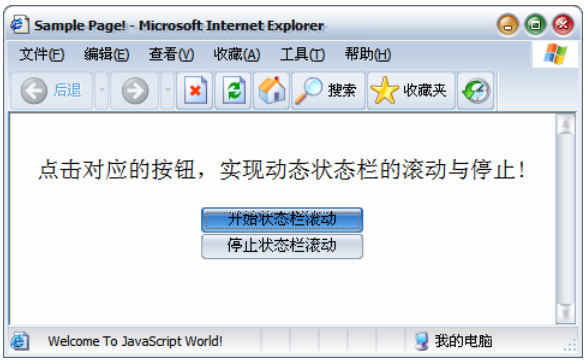

图 7.20 状态栏信息左右滚动显示

单击“停止状态栏滚动”按钮后，状态栏信息滚动显示的效果停止，状态栏显示“The Moving Status have been stopped!”，如图 7.21 所示。

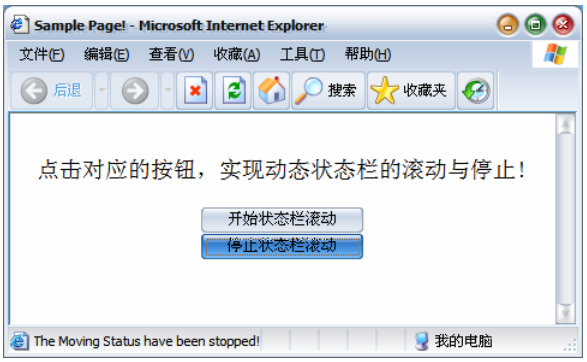

图 7.21 状态栏滚动停止

该实例演示了 Window 对象的 setInterval()和 clearInterval()方法的使用情况，如果要更改滚动的速度，只需修改 TimerID=setInterval("startStatus();",100)语句里面的间隔时间“100ms”即可实现。

7.2.3 事件超时控制

Window 对象提供 setTimeout()方法用于设置某事件的超时，即在设定的时间到来时触发某指定的事件，该方法的实际应用有警告框的显示时间和状态栏的跑马灯效果、打字效果等。其语法如下：

```
var timer=window.setTimeout(targetProcess,itime);
var timer=setTimeout(targetProcess,itime);
```

参数 targetProcess 表示设定超时的目标事件，参数 itime 表示设定的超时时间，以毫秒(ms)为单位，返回值 timer 为该事件超时的引用变量名。

同时，Window 对象提供 clearTimeout()方法来清除通过参数传入的事件超时操作。该语法如下：

```
clearTimeout(timer);
```

该方法接受唯一的参数 timer 指定要清除的事件超时引用变量名，方法执行后将该事件

超时设置为失效。考察如下演示状态栏打字效果的代码：

```
//源程序 7.3
<!DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page!</title>
<script language="JavaScript" type="text/javascript">
<!--
var seq=0;
var TimerID;
//超时时间
var interval = 120;
//设定要实现状态栏打字效果的字符串
var MyStatus = "Welcome to JavaScript World!";
var strLength= MyStatus.length;
//设定状态栏的显示内容
function MyScroll()
{
   window.status=MyStatus.substring(0, seq+1);
   seq++;
   if(seq>= strLength)
   {
      seq=0;
      window.status='';
      //设定触发事件的时间间隔
      TimerID=setTimeout("MyScroll();",interval);
   }
   else
      TimerID=setTimeout("MyScroll();",interval);
}
//停止该超时操作
function MyStopScroll()
{
   clearTimeout(TimerID);
   window.status="The Changing Status have been stopped!";
}
//-->
</script>
</head>
<body onload="window.status='Original Status!'">
<br>
<center>
<p>单击对应的按钮，实现状态栏打字效果的显示与停止!</p>
<form name="MyForm">
   <input type="button" value="开始状态栏滚动" onclick="MyScroll()"><br>
   <input type="button" value="停止状态栏滚动" onclick="MyStopScroll()"><br>
</form>
</center>
</body>
</html>
```

程序运行后，出现如图 7.19 所示页面。在此页面单击“开始状态栏滚动”按钮后，状

态栏中状态信息从空白逐字增加直到指定字符串结束，然后清空为空白并重复此过程，如图 7.22 所示。

图 7.22 状态栏的打字效果

单击“停止状态栏滚动”按钮后，状态栏的打字效果停止，并显示状态信息“The Changing Status have been stopped!”，如图 7.23 所示。

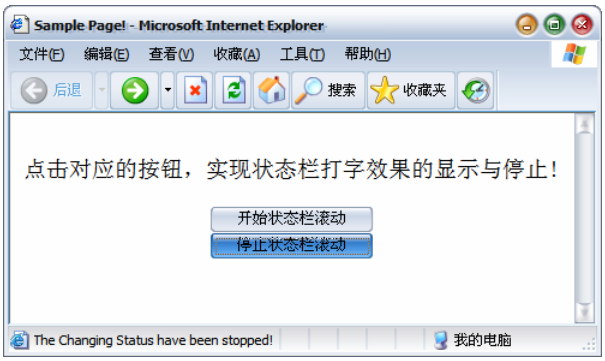

图 7.23 状态栏的打字效果失效

该实例演示了 Window 对象的 setTimeout()和 clearTimeout()方法的使用情况，如果要更改打字效果中字符出现的速度，只需修改 TimerID=setTimeout("MyScroll();",interval)语句里面的间隔时间 interval 即可实现。

7.2.4 创建和管理新窗口

Window 对象提供完整的方法用于创建新窗口并在父窗口与字窗口之间进行通信。一般来说，主要使用其 open()方法创建新浏览器窗口，新窗口可以包含已存在的 HTML 文档或者完全由该方法创建的新文档，其语法如下：

```
var newWindow=window.open(targetURL,pageName, options,repalce);
var newWindow=open(targetURL,pageName, options,repalce);
```

其中参数：

- targetURL：指定要打开的目标文档地址；
- pageName：设定该页面的引用名称；
- options：指定该窗口的属性，如页面大小、有否工具条等。

其中 options 包含一组用逗号隔开的可选属性对，用以指明该窗口所具备的各种属性，

其属性及对应的取值如表 7.1 所示。

表 7.1 options参数可包含的属性

属性	取值	简要说明
directories	yes/no	目标窗口是否具有目录按钮
height	integer	目标窗口的高度
left	interger	目标窗口与屏幕最左边的距离
location	yes/no	目标窗口是否具有地址栏
menubar	yes/no	目标窗口是否具有菜单栏
resizable	yes/no	目标窗口是否允许改变大小
scrollbars	yes/no	目标窗口是否具有滚动条
status	yes/no	目标窗口是否具有状态栏
toolbar	yes/no	目标窗口是否具有工具栏
top	integer	目标窗口与屏幕最顶端的距离
width	integer	目标窗口的宽度

注意：left、height、top、width 属性的取值为整数，为像素值。其余取值为 yes/no，分别表示目标具有或不具有某种属性。在当前浏览器版本中，可用 1 代替 yes，用 0 代替 no。

窗口建立后，可通过新窗口的 document 对象的 write()方法往该窗口写入内容，可以是纯粹的字符串，也可是 HTML 格式的字符串，后者将被浏览器解释之后再显示。

操作完成后，可通过 Window 对象的 close()方法来关闭该窗口，close()方法的语法如下：

```
windowName.close();
```

使用 close()方法关闭某窗口之前，一定要核实该窗口是否已经定义、是否已经定义，如果目标窗口未定义或已经被关闭，则 close()方法返回错误信息。

考察如下的综合实例，该实例演示了如何新建浏览器窗口及往该窗口写入内容的方法：

```
//源程序 7.4
<!DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page!</title>
<script language="JavaScript" type="text/javascript">
<!--
var ExistWindow;
var NewWindow;
var strOptions="";
//产生窗口属性字符串
function CreateStr()
{
  var CheckLength=document.MyForm.elements.length;
  for(var i=0;i<CheckLength-1;i++)
  {
    if((document.MyForm.elements[i].type=="checkbox")&&
        (document.MyForm.elements[i].checked))
      strOptions+=document.MyForm.elements[i].name+"=yes,";
  }
  strOptions+="height="+document.MyForm.height.value+",";
  strOptions+="width="+document.MyForm.width.value+",";
  strOptions+="top="+document.MyForm.top.value+",";
  strOptions+="left="+document.MyForm.left.value;
}
```

```
//新建已存在的窗口
function CreateExistWindow()
{
   strOptions="";
   CreateStr();
   var MyURL=document.MyForm.MyURL.value;
   var MyName=document.MyForm.MyName.value;
   ExistWindow=window.open(MyURL,MyName,strOptions);
   if(!window.ExistWindow)
      alert("The target window is not exist!");
}
//新建空白窗口并写入预设的字符串
function WriteNewWindow()
{
   strOptions="";
   CreateStr();
   var MyURL='';
   var MyName=document.MyForm.MyName.value;
   NewWindow=window.open(MyURL,MyName,strOptions);
   if(window.NewWindow)
   {
      NewWindow.document.write(document.MyForm.MyContents.value);
      NewWindow.focus();
   }
   else
      alert("The window to be written is not exist!");
}
//关闭当前子窗口
function CloseWindow()
{
   if(window.ExistWindow)
      ExistWindow.close();
   else if(window.NewWindow)
      NewWindow.close();
   else
      alert("The window to be closed is not exist!");
}
//-->
</script>
</head>
<body>
<center>
设置新窗口属性,然后单击“确定”按钮产生新窗口!
<hr>
<form name="MyForm">
   窗口地址 ： <input type="text" name="MyURL" id="MyURL" value="temp.html"><br>
   窗口名称 ： <input type="text" name="MyName" id="MyName" value="MySamplePage"><br>
   窗口高度 ： <input type="text" name="height" id="height" maxlength=4 value=290><br>
   窗口宽度 ： <input type="text" name="width" id="width" maxlength=4 value=324><br>
   左边距离 ： <input type="text" name="top" id="top" maxlength=4 value=100><br>
   顶端距离 ： <input type="text" name="left" id="left" maxlength=4 value=100><br>
   窗口内容 ： <textarea name="MyContents" id="MyContents" rows=4 cols=19>
               Contents Write to the new blank window!</textarea><br>
```

```
  滚动条    ：<input type="checkbox" name="scrollbar" id="scrollbar"><br>
  大小改变 ：<input type="checkbox" name="resizable" id="resizable"><br>
  目录按钮 ：<input type="checkbox" name="directories" id="directories"><br>
  地址栏    ：<input type="checkbox" name="location" id="location"><br>
  菜单栏    ：<input type="checkbox" name="menubar" id="menubar"><br><hr>
  <input type="button" value="新建浏览器窗口" onclick="CreateExistWindow()">
  <input type="button" value="写内容进空白窗口" onclick="WriteNewWindow()">
  <input type="button" value="关闭当前子窗口" onclick="CloseExistWindow()">
</form>
</center>
</body>
</html>
```

程序运行后，出现如图 7.24 所示的页面。

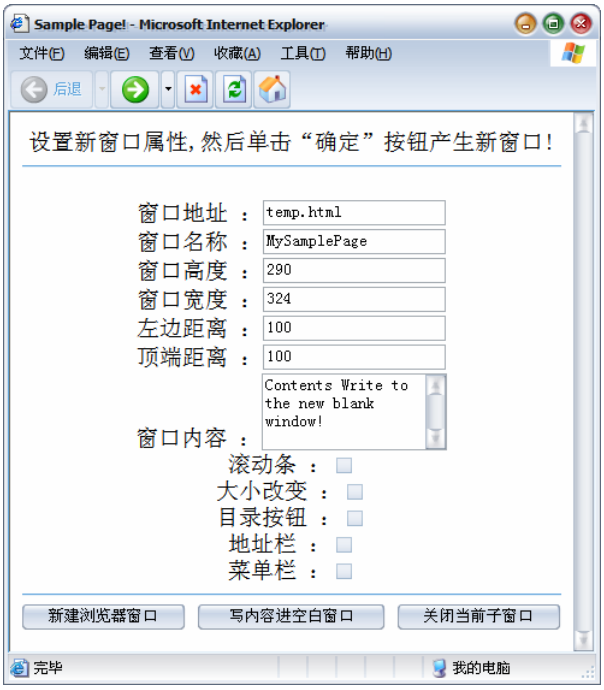

图 7.24 程序运行后的原始页面

在原始页面选中五个 checkbox 单选框，单击“新建浏览器窗口”按钮，浏览器将按照表单中设定的属性弹出当前目录下 HTML 文档“temp.html”的页面，如图 7.25 所示。

图 7.25 按照设定的属性加载已存在的 HTML 文档

在原始页面选中五个 checkbox 单选框，在原始页面单击“写内容进空白页面”按钮，浏览器将按照表单中设定的属性生成新浏览器窗口，并将文本域“MyContents”的内容“Contents Write to the new blank window!”写进该页面，如图 7.26 所示。

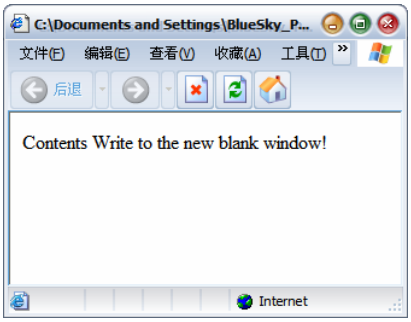

图 7.26 生成新浏览器页面并写入指定的内容

该新建的 HTML 页面拥有写入的字符串，不包含任何格式信息。查看源文件，仅包含“Contents Write to the new blank window!”字符串。

当然，往新建的 HTML 窗口动态写入 HTML 语句也是可行的。查看如下代码：

```
var embedHtml="<html><head><title>Sample Page!</title></head>";
embedHtml+="<body>Contents Write to the new blank window!</body></html>";
NewWindow.document.write(embedHtml);
```

将上述代码替换源程序 7.4 中的下列语句：

```
NewWindow.document.write(document.MyForm.MyContents.value);
```

保存后，运行该程序，单击“写内容进空白页面”按钮，弹出新页面，与源程序 7.4 一样，但查看源程序，如图 7.27 所示。

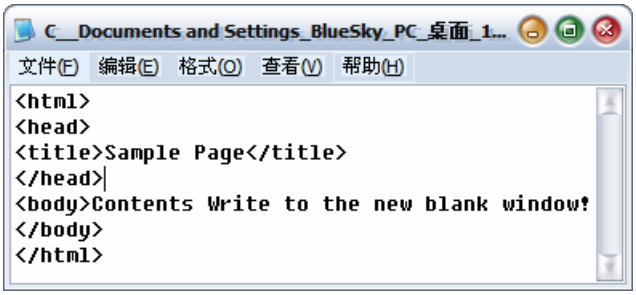

图 7.27 将 HTML 语句写入新窗口

通过上述实例，读者可以适当扩展，按照需要给新窗口适当的属性值，然后根据新窗口的引用名称动态写入 HTML 语句生成交互性较强的页面。

7.2.5 常见属性和方法汇总

Window 对象提供诸多属性和方法用于浏览器窗口操作，如获取和设置当前窗口信息、创建浏览器窗口等。但由于各大浏览器厂商在继承 DOM 标准的基础上各自扩展了 Window 对象，而且浏览器的版本对 Window 对象的支持程度也不一样。出于兼容性考虑，表 7.2 列出了 Internet Explorer（简称 IE，下同）和 Netscape Navigator（简称 NN，下同）浏览器平

台通用的 Window 对象常见属性和方法。

表 7.2 Window对象常见属性和方法汇总

类型	项目	简要说明
属性	closed	表示窗口是否已被关闭
	defaultStatus	窗口底部默认的状态栏信息
	document	窗口中当前文档对象
	frames	包含窗口中所有Frame对象的数组
	history	包含窗口历史URL清单的History对象
	location	包含与Window对象相关联的URL地址的对象
	name	当前窗口的标识
	opener	表示打开窗口的Window对象
	parent	与包含某个窗口的父窗口含义相同
	self	与当前窗口的含义相同
	status	窗口底部的状态栏信息
	top	指一组嵌套窗口的最上层浏览器窗口
方法	alert()	显示提示信息对话框
	blur()	使当前窗口失去焦点
	clearInterval(TimerID)	使由参数TimerID指定的间隔定时器失效
	clearTimeout(TimerID)	使由参数TimerID指定的超时设置失效
	close()	关闭当前窗口
	conform(text)	显示确认对话框，text为确认内容
	focus()	使当前窗口获得焦点
	moveBy(deltaX,deltaY)	将浏览器窗口移动到由参数deltaX和deltaY(像素)指定相对距离的位置
	moveTo(x,y)	将浏览器移动到由参数x和y(像素)指定的位置
	open(URL,Name,Options)	按照Options指定的属性打开新窗口并创建Window对象
	prompt(text[, str])	显示提示对话框，text为问题，str为默认答案（可选参数）
	resizeBy(deltaX,deltaY)	将浏览器窗口大小按照参数deltaX和deltaY(像素)指定的相对像素改变
	resizeTo(x,y)	将浏览器窗口的大小按照参数参数x和y(像素)指定的值进行设定
	scroll(hori,Verti)	将目标文档移动到浏览器窗口中由参数hori和Verti指定的位置（NN3+）
	scrollBy(deltaX,deltaY)	在浏览器窗口中将文档移动由deltaX和deltaY指定相对距离的位置
	scrollTo(x,y)	在浏览器窗口中将文档移动到由x和y指定的位置
	setInterval(expression, milliseconds, [arguments])	通过由参数milliseconds指定的时间间隔重复触发由参数expression指定的表达式求值或函数调用，可选参数arguments为供函数调用的参数列表，以逗号为分隔符
	setTimeout(expression, milliseconds, [arguments])	通过由参数milliseconds指定的超时时间触发由参数expression指定的表达式求值或函数调用，可选参数arguments为供函数调用的参数列表，以逗号为分隔符

Window 对象为 Web 应用开发者提供了丰富的属性和操作浏览器窗口及事件的方法，通过 Window 对象，可以访问对象关系层次中处于其下层的对象如 Navigator 对象等。下面讨论与浏览器紧密相关的 Navigator 对象。

7.3 Navigator 对象

由于浏览器及其版本对 JavaScript 脚本的支持情况不同，出于脚本代码兼容性考虑，经常需要获取客户端浏览器相关信息，以根据其浏览器具体情况编制不同的脚本代码。

Navigator 对象最初由 Netscape 浏览器引入，并在其 NN2 中获得支持。Microsoft 在其 IE3 上引入 Navigator 对象，但只支持其部分属性和方法。由于 Navigator 对象为程序员提供了十分有效的浏览器相关信息而得到较为广泛的应用，Microsoft 在其 IE4 中引入 Navigator 对象的克隆版本即 clientInformation 对象并在 IE4 后续版本中得到更为完善的支持，该对象的所有属性和方法与 Navigator 对象完全相同。不同的是，clientInformation 对象仅适用于 IE

浏览器，而 Navigator 对象则适用于所有浏览器，当然也包括 IE 浏览器。

同 Window 对象一样，Navigator 对象为浏览器对象模型中的顶级对象，而不是作为其他对象的属性而存在。相比较 Window 对象而言，Navigator 对象与浏览器及其版本的关联程度更紧密，对编写代码兼容性较强的应用程序贡献更大，但 Navigator 对象的属性多为只读，且提供的操作方法也较少。

7.3.1 获取浏览器信息

在编写跨平台 JavaScript 脚本时，须事先获取客户端浏览器的相关信息，然后作对应的操作，此方法可解决脚本代码在各浏览器中的兼容性问题。下面的代码演示了如何获取客户端浏览器的相关信息并作相应的处理：

```
//源程序 7.5
<!DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page!</title>
<script language="JavaScript" type="text/javascript">
<!--
//根据客户端浏览器信息确定脚本的走向
function GetInfo()
{
  //判断浏览器类型，若浏览器为 NN，则转向 displayNNInfo()
  if(navigator.appName=="Netscape")
    displayNNInfo();
  //若浏览器为 IE,则转向 displayIEInfo()
  else if(navigator.appName=="Microsoft Internet Explorer")
    displayIEInfo();
  //否则,输出警告信息
  else
    alert("对不起,当前浏览器类型不支持!");
}
//如果客户端浏览器为 NN,则触发此函数
function displayNNInfo()
{
  var msg="\nNN 浏览器信息 : \n\n"+"检测结果 : \n\n"+"通用属性 : \n";
  msg+="----appCodeName : "+navigator.appCodeName+"\n";
  msg+="----appName : "+navigator.appName+"\n";
  msg+="----appVersion : "+navigator.appVersion+"\n";
  msg+="----cookieEnabled : "+navigator.cookieEnabled+"\n";
  msg+="----mimeTypes.length : "+navigator.mimeTypes.length+"\n";
  msg+="----plugins.length : "+navigator.plugins.length+"\n";
  msg+="----platform : "+navigator.platform+"\n";
  mag+="----userAgent : "+navigator.userAgent+"\n";
  msg+="扩展属性 : \n";
  msg+="----language : "+navigator.language+"\n";
  alert(msg);
}
//如果客户端浏览器为 IE,则触发此函数
```

```
function displayIEInfo()
{
   var msg="\nIE 浏览器信息 : \n\n"+"检测结果 : \n\n"+"通用属性 : \n";
   msg+="----appCodeName : "+navigator.appCodeName+"\n";
   msg+="----appName : "+navigator.appName+"\n";
   msg+="----appVersion : "+navigator.appVersion+"\n";
   msg+="----cookieEnabled : "+navigator.cookieEnabled+"\n";
   msg+="----mimeTypes.length : "+navigator.mimeTypes.length+"\n";
   msg+="----plugins.length : "+navigator.plugins.length+"\n";
   msg+="----platform : "+navigator.platform+"\n";
   msg+="----userAgent : "+navigator.userAgent+"\n";
   msg+="扩展属性 : \n";
   msg+="----appMinorVersion : "+navigator.appMinorVersion+"\n";
   msg+="----cpuClass : "+navigator.cpuClass+"\n";
   msg+="----language : "+navigator.language+"\n";
   msg+="----browserLanguage : "+navigator.browserLanguage+"\n";
   msg+="----userLanguage : "+navigator.userLanguage+"\n";
   msg+="----systemLanguage : "+navigator.systemLanguage+"\n";
   msg+="----onLine : "+navigator.onLine+"\n";
   msg+="----userProfile : "+navigator.userProfile+"\n";
   alert(msg);
}
//-->
</script>
</head>
<body>
<hr>
<center>
<form name="MyForm">
   <input type="button" value="测试" onclick="GetInfo()">
</form>
</center>
</body>
</html>
```

程序运行后，单击页面中的“测试”按钮，如果客户端浏览器为 IE4+，则弹出警告框如图 7.28 所示。

图 7.28 当客户端浏览器为 IE4+时弹出警告框

如果客户端浏览器为 NN2+，则弹出警告框如图 7.29 所示。

图 7.29 当客户端浏览器为 NN2+时弹出提示框

浏览器载入页面后，客户单击“测试”按钮，触发 GetInfo()函数，该函数判断当前浏览器的类型：若其 appName 属性为“Netscape”，则转向 displayNNInfo()函数输出当前浏览器相关信息；若其 appName 属性为“Microsoft Internet Explorer”，则转向 displayIEInfo()函数输出当前浏览器相关信息。

> 注意：上述程序的调试平台为 IE6 和 NN7，较低浏览器版本可能不支持某些属性，有关 Navigator 对象属性的版本支持将在后面列表详述。

7.3.2 常见方法和属性汇总

Navigator 对象拥有的属性和方法随浏览器版本的更新而不断增加，总的来说，除了早期的 Navogator 对象的基本属性和方法之外，大多数新增的属性和方法都与浏览器版本相关。表 7.3 列出了 Navigator 常见的属性、方法及浏览器支持情况。

表 7.3 Navigator常见的属性和方法

类型	项目	简要说明	浏览器支持
属性	appCodeName	返回包含每个浏览器的客户类，代表浏览器代码号	NN2+、IE3+
	appName	返回浏览器官方名称，如IE的Microsoft Internet Explorer等	NN2+、IE3+
	appVersion	返回浏览器的版本号	NN2+、IE3+
	cookieEnabled	标记浏览器的cookie功能是否已开启	NN6+、IE4+
	MimeTypes	保存MIME类型信息的数组	NN4+、IE4+
	Plugins	保存网页中插件程序的数组	NN3+、IE4+
	platform	保存操作系统的类型	NN4+、IE4+
	userAgent	保存从客户端向服务器发送的HTTP协议用户代理头的值	NN2+、IE3+
	appMinorVersion	返回浏览器的次版本号	IE4+
	cpuClass	返回运行浏览器的中央处理器种类	IE4+
	browserLanguage	返回程序本地语言版本的标识符	IE4+
	userLanguage	返回当前操作系统使用的自然语言	IE4+
	systemLanguage	返回操作系统默认使用的语言	IE4+
	online	标记浏览器确定脱机浏览的状态，联机状态下该属性为true	IE4+
	userProfile	返回用户的档案设置	IE4+
	language	返回浏览器应用程序的语言代码	NN4+
方法	javaEnabled()	返回浏览器是否禁止Java的标志位	NN3+，IE4+
	taintEnabled()	返回浏览器支持数据感染安全特性的标志位	NN3+、IE4+
	preference()	允许标识的脚本获取并设置某些 Navigator 的首选项信息	NN4+

Navigator 对象的属性和方法在实际调用过程中，除了理解其基本含义外，还需了解以下几项内容：

- appName 属性返回浏览器应用程序的官方名称，IE 浏览器的官方名称为“Microsoft Internet Explorer”，NN 浏览器的官方名称为“Netscape”，而有些浏览器则通过 Navigator 对象的扩展方法来检测其官方名称，如 Opera 浏览器的 isOpera()、Safari 浏览器的 isSafari()等。
- appVersion 属性返回当前浏览器的版本号，一般情况下可通过 parseInt()和 parseFloat()方法提取其中的数值再进行相关比较，但此数值更多的是表现浏览器版本的继承特性而不是真正的版本号，如 IE6 的 appVersion 属性在提取数值后，返回 4 而不是 6。
- platform 属性返回操作系统的类型。Win32 代表 Window XP、Win98 代表 Windows 98、WinNT 代表 Windows NT、Win16 代表 Window3.x、Mac68k 代表 Mac(680x0 CPU)、MscPPC 代表 Mac(PowerPC CPU)、SunOS 代表 Saloris 等。
- plugins 属性在 IE4+上获得支持，但返回一个空数组，表示不包含任何 IE 中不存在的对象。

Navigator 对象从本质上说是在顶级对象模型中与浏览器类型及其版本紧密相关的顶级对象，其属性和方法随着浏览器类型、版本及系统设置的完成而确定下来，多为只读的属性和方法。Navigator 对象又包括两个作为其属性的对象，分别为 mimeType 对象和 plugin 对象，该部分内容在后续章节详细叙述。下面介绍与浏览器物理相关的 Screen 对象。

7.4 Screen 对象

Screen 对象最初由 NN4 引入，该对象提供了客户端用户屏幕的相关信息，如屏幕尺寸、颜色深度等。如同 Navogator 对象由 NN2 引入后 Micorsoft 在其 IE3 中引入 Navigator 对象的克隆版本 clientInformation 对象一样，在 NN4 中 Screen 对象引入后，Microsoft 在其 IE4 中定义了新的 Screen 对象，其属性和方法与 NN4 中定义的完全相同，不同点在于 IE4 中的 Screen 对象作为 Window 对象的属性而存在，而 NN4 中 Screen 对象和 Window 对象同为顶级对象模型的成员。

Screen 对象的属性可用 screen.property 的方式调用，在 IE4+中还可以使用如下的语法：

```
[window.]navigator.screen.property
```

Screen 对象基本的属性包括 height、width 和 colorDepth 等，但各浏览器厂商都对其进行了一定的扩展，如 NN4 扩展了 availLeft 和 availTop、pixelDepth 等属性，用于返回屏幕可用区域的初始像素位置坐标（像素）；IE4 扩展了 bufferDepth 属性，用于打开 offscreen 缓冲并控制缓冲的颜色深度等。除此之外，NN4 和 IE4 共同扩展了 availHeight 和 availWidth 等属性，前两个表示客户端屏幕的可用尺寸（像素），而后者返回客户端的“显示”控制面板中设置的颜色位数。

7.4.1 获取客户端屏幕信息

在 Web 应用程序中，为某种特殊目的如固定文档窗口相对于屏幕尺寸的比例、根据显示器的颜色位数选择需要加载的目标图片等都需要首先获得屏幕的相关信息。Screen 对象提供了 height 和 width 属性用于获取客户屏幕的高度和宽度信息，如分辨率为 1024*768 的显示器，调用这两个属性后分别返回 1024 和 768。但并不是所有的屏幕区域都可以用来显示

文档窗口，如任务栏等都占有一定的区域。为此，Screen 对象提供了 availHeight 和 availWidth 属性来返回客户端屏幕的可用显示区域。一般来说，Windows 操作系统的任务栏默认在屏幕的底部，也可以被拖动到屏幕的两侧或者顶部。假定屏幕的分辨率为 1024*768，当任务栏在屏幕的底部或者顶部时，其占据的屏幕区域大小为 1024*30（像素）；当任务栏被拖动到屏幕两侧时，其占据的屏幕区域大小为 60*768。

考察如下针对不同浏览器平台获取客户端屏幕相关信息的代码：

```
//源程序 7.6
<!DOCTYPE HTML PUBLIC   "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page!</title>
<script language="JavaScript" type="text/javascript">
<!--
//根据客户端浏览器信息确定脚本的走向
function CheckOS()
{
   //判断浏览器类型
   //若浏览器为 NN,则转向 displayNNInfo()
   if(navigator.appName=="Netscape")
      GetWindowNN();
   //若浏览器为 IE,则转向 displayIEInfo()
   else if(navigator.appName=="Microsoft Internet Explorer")
      GetWindowIE();
   //否则,输出警告信息
   else
      alert("对不起,当前浏览器类型不支持!");
}
//如果客户端浏览器为 NN,则触发此函数
function GetWindowNN()
{
   var msg="\n 屏幕信息(NN4+) : \n\n";
   msg+="通用属性 : \n";
   msg+="----availHeight : "+screen.availHeight+"\n";
   msg+="----availWidth : "+screen.availWidth+"\n";
   msg+="----Height : "+screen.height+"\n";
   msg+="----Width : "+screen.width+"\n";
   msg+="----colorDepth : "+screen.colorDepth+"\n\n";
   msg+="扩展属性 : \n";
   msg+="----availLeft : "+screen.availLeft+"\n";
   msg+="----availTop : "+screen.availTop+"\n";
   msg+="----pixelDepth : "+screen.pixelDepth+"\n";
   alert(msg);
}
//如果客户端浏览器为 IE,则触发此函数
function GetWindowIE()
{
   var msg="\n 屏幕信息(IE4+) : \n\n";
   msg+="通用属性 : \n";
   msg+="----availHeight : "+screen.availHeight+"          \n";
   msg+="----availWidth : "+screen.availWidth+"\n";
```

```
    msg+="----Height : "+screen.height+"\n";
    msg+="----Width : "+screen.width+"\n";
    msg+="----colorDepth : "+screen.colorDepth+"\n\n";
    msg+="扩展属性 : \n";
    msg+="----bufferDepth : "+screen.bufferDepth+"\n";
    alert(msg);
}
//-->
</script>
</head>
<body>
<hr>
<center>
<form name="MyForm">
    <input type="button" value="测试" onclick="CheckOS()">
</form>
</center>
</body>
</html>
```

保存文档，若使用 IE 浏览器打开，当任务栏在屏幕底端或顶端时，在目标页面上鼠标鼠标单击“测试”按钮，将弹出警告框如图 7.30 所示。

图 7.30 IE 中当任务栏在底端或顶部时的屏幕信息

当任务栏在屏幕两侧时，在目标页面上鼠标单击“测试”按钮，将弹出警告框如图 7.31 所示。

图 7.31 IE 中当任务栏在两侧时的屏幕信息

若使用 NN 浏览器打开，无论任务栏在屏幕的哪个位置，在目标页面上鼠标单击“测试”按钮，将弹出警告框如图 7.32 所示。

图 7.32 NN 中获取屏幕的相关信息

可以发现，不管任务栏在何种位置，其 availHeight 和 availWidth 属性都返回默认状态栏在底部时可用的区域高度和宽度。

获取客户端屏幕的相关信息后，就可以通过 JavaScript 脚本来进行感兴趣的操作，如窗口的尺寸、位置及文档中图片的大小等。典型应用如全屏页面、保持图片与文档页面大小比例等。

7.4.2 定位窗口到指定位置

通过 Screen 对象的属性获得屏幕的相关信息后，结合 Window 对象有关窗口移动、更改尺寸的属性，可准确定位目标窗口。实际应用中如跟随鼠标移动的窗口、拥有固定位置的窗口等。考察如下控制窗口位置的代码：

```
//源程序 7.7
<!DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page!</title>
<script language="JavaScript" type="text/javascript">
<!--
//设定窗口水平和垂直位置的参数(像素)
var ix;
var iy;
//设定窗口宽度和高度的参数(像素)
var iwidth;
var iheight;
//设定窗口移动的方向和速度的参数
```

```
var xDirection=7;
var yDirection=4;
//设定与窗口移动相关联的间隔时间计时器
var TimerID;
//初始化窗口大小和位置
function InitWindow()
{
   ix=200;
   iy=200;
   iwidth=480;
   iheight=320;
   window.moveTo(ix,iy);
   window.resizeTo(iwidth,iheight);
}
//将浏览器窗口居中放置
function CenterWindow()
{
   var ix=(screen.width-iwidth)/2;
   var iy=(screen.height-iheight)/2;
   window.moveTo(ix,iy);
}
//设定间隔时间计时器
function StartMove()
{
   //此处的 100 代表改变的速度,即 100ms 改变一次位置
   TimerID=setInterval("MoveWindow();",100);
}
//控制页面移动
function MoveWindow()
{
   //如果目标页面水平方向在屏幕之内,则不改变水平移动方向
   if((ix+iwidth<screen.width)&&(ix>0))
   {
      ix=ix+xDirection;
   }
   //如果目标页面水平方向在屏幕之外,则改变水平移动方向
   if((ix+iwidth>=screen.width)||(ix<=0))
   {
      xDirection=-xDirection;
      ix=ix+xDirection;
   }
   //如果目标页面垂直方向在屏幕之内,则不改变垂直移动方向
   if((iy+iheight<screen.height-30)&&(iy>0))
   {
      iy=iy+yDirection;
   }
   //如果目标页面垂直方向在屏幕之外,则改变垂直移动方向
   if((iy+iheight>=screen.height-30)||(iy<=0))
   {
      yDirection=-yDirection;
      iy=iy+yDirection;
   }
   //移动窗口至指定的位置
```

```
   window.moveTo(ix,iy);
}
//停止窗口的滚动,回到原始位置
function StopMove()
{
   clearInterval(TimerID);
   CenterWindow();
}
//全屏化浏览器窗口
function FullWindow()
{
   //判断浏览器类型
   //若浏览器为 NN
   if(navigator.appName=="Netscape")
   {
      ix=screen.availLeft;
      iy=screen.availTop;
      iwidth=screen.availWidth;
      iheight=screen.availHeight;
      window.moveTo(ix,iy);
      window.resizeTo(iwidth,iheight);
   }
   //若浏览器为 IE
   else if(navigator.appName=="Microsoft Internet Explorer")
   {
      ix=0;
      iy=0;
      iwidth=screen.Width;
      iheight=screen.Height;
      window.moveTo(ix,iy);
      window.resizeTo(iwidth,iheight);
   }
   //否则,输出警告信息
   else
      alert("对不起,当前浏览器类型不支持!");
}
//-->
</script>
</head>
<body>
<center>
<P>顺序单击如下按钮,实现窗口准确定位<p>
<form name="MyForm">
   <input type="button" value="初始化浏览器窗口" onclick="InitWindow()"><br>
   <input type="button" value="将浏览器窗口居中" onclick="CenterWindow()"><br>
   <input type="button" value="开始目标窗口移动" onclick="StartMove()"><br>
   <input type="button" value="停止目标窗口移动" onclick="StopMove()"><br>
   <input type="button" value="全屏化浏览器窗口" onclick="FullWindow()"><br>
</form>
</center>
</body>
</html>
```

程序运行后，原始页面如图 7.33 所示。

图 7.33 使用 Screen 对象的属性控制窗口位置

程序主要分为如下几个步骤：

- 在页面中单击“初始化浏览器窗口”按钮，按照 InitWindow()函数设定的参数值初始化目标窗口。通过 Window 对象的 moveTo()方法将窗口移动到(200,200)位置，并通过其 resizeTo()方法改变目标窗口大小为 480*320，单位均为像素值；
- 单击“将浏览器窗口居中”按钮，JavaScript 脚本通过 Screen 对象的 width 和 height 属性及窗口的宽度和高度计算窗口居中时其左上顶点的坐标，通过 Window 对象的 moveTo()方法将目标窗口居中；
- 单击“开始目标窗口移动”按钮，触发 StartMove()函数启动间隔时间计时器，该计时器连接到函数 MoveWindow()，后者计算目标窗口的当前位置设定移动的方向和下一个位置，并通过 Window 对象的 moveTo()方法更新目标窗口的位置。此过程在间隔时间计时器的控制下循环进行，实现窗口的移动；
- 单击“停止目标窗口移动”按钮，取消控制窗口移动的间隔时间计时器，并通过 CenterWindow()函数将窗口居中放置；
- 单击“全屏化浏览器窗口”按钮，触发 FullWindow()函数，后者通过检查客户端浏览器的类型，并调用其支持的属性，通过 Window 对象的 moveTo()和 resizeTo()方法将目标窗口最大化。

上述代码演示了如何精确控制指定窗口的位置，在窗口移动过程中，可通过缩短间隔计时器时间并减小像素的每次偏移 xDirection 和 yDirection 的初始值来达到使目标窗口移动流畅的目的。

7.4.3 常见属性和方法汇总

Screen 对象提供较少但非常有效的属性用于获取客户端屏幕的相关信息并据此作出相应的动作。表 7.4 列出了 Screen 对象常见的属性及浏览器支持情况。

表 7.4 Screen对象常见属性和方法汇总

属性	简要说明	浏览器支持
availHeight	返回客户端屏幕分辨率中可用的高度（像素）	NN4+、IE4+
availWIdth	返回客户端屏幕分辨率中可用的宽度（像素）	NN4+、IE4+
height	返回客户端屏幕分辨率中的高度（像素）	NN4+、IE4+
width	返回客户端屏幕分辨率中的宽度（像素）	NN4+、IE4+
colorDepth	返回客户端“显示”控制面板中设置的颜色位数	NN4+、IE4+
availLeft	返回客户端屏幕的可用区域最左边初始像素位置（像素）	NN4+
availTop	返回客户端屏幕的可用区域最顶端初始像素位置（像素）	NN4+
pixelDepth	返回客户端“显示”控制面板中设置的颜色位数	NN4+

bufferDepth	返回标记offscreen缓冲是否打开和缓冲的颜色深度，默认值为0	IE4+

Screen 对象保存了客户端屏幕的相关信息，与文档本身相关程度较弱。下面介绍在顶级对象模型中与浏览器浏览网页后保存已访问页面和所在位置相关信息的 History 对象和 Location 对象。

7.5 History 对象

在顶级对象模型中，History 对象处于 Window 对象的下一个层次，主要用于跟踪浏览器最近访问的历史 URL 地址列表，但除了 NN4+中使用签名脚本并得到用户许可的情况之外，该历史 URL 地址列表并不能由 JavaScript 脚本显示读出，而只能通过调用 History 对象的方法模仿浏览器的动作来实现访问页面之间的漫游。

7.5.1 使用 back()和 forward()方法进行站点导航

History 对象提供 back()、forward()和 go()方法来实现站点页面的导航。back()和 forward()方法实现的功能分别与浏览器工具栏中“后退”和“前进”导航按钮相同，而 go()方法则可接受合法参数，并将浏览器定位到由参数指定的历史页面。这三种方法触发脚本检测浏览器的历史 URL 地址记录，然后将浏览器定位到目标页面，整个过程与文档无关，但在 IE 和 NN 浏览器中这两种方法又存在着不同点。

在 IE3+中，back()和 forward()方法模拟工具栏的物理行为，并不局限于框架集中特定的框架。如果要确保框架集中特定框架的跨浏览器的行为，而不是跨浏览器版本的行为，则须将 history.back()和 history.forward()方法的引用传递给父窗口。

在 NN4 中，History 对象的 back()和 forward()方法遵循其基本功能，同时又变成一对 Window 对象的方法 window.back()和 window.forward()。除此之外，History 对象不局限于框架集中使用，当框架集中使用 parent.frameName.history.forward()到达框架历史 URL 地址记录的尾部时，此方法无效。

站点导航是 back()和 forward()方法应用最为广泛的场合，可以想象在没有工具栏或菜单栏的页面（如用户注册进程中间页面等）中设置导航按钮的必要性。考察站点页面导航的实例，先看简单的框架集文档“index.html”的代码：

```
//源程序 7.8
<html>
<head>
   <title>Sample Page!</title>
</head>
<frameset cols="50%,50%">
   <frame name="MainBoard" src="left_main.html">
   <frame name="Display" src="01.html">
</frameset>
</html>
```

其中 HTML 文档“01.html”与下面将要引用的“02.html”、“03.html”、“04.html”文档类似，只列出“01.html”文档的代码：

```
//源程序 7.9
<!DOCTYPE HTML PUBLIC   "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
```

```
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>测试文档 01</title>
</head>
<body>
<center>
<p>测试文档 01</p>
</center>
<p>测试文档具体内容</p>
</body>
</html>
```

页面中左框架文档“left_main.html”的代码如下：

```
//源程序 7.10
<!DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page!</title>
</head>
<body>
<center>
<p><b>控制框架页面</b></p>
<p>
  <a href="01.html" target="Display">演示文档 01</a><br>
  <a href="02.html" target="Display">演示文档 02</a><br>
  <a href="03.html" target="Display">演示文档 03</a><br>
  <a href="04.html" target="Display">演示文档 04</a><br>
</p>
<form name="MyForm">
  <p><b>NN4+独有方法 : </b></p>
  window.back() : <input type=button name="MyBack" value="返回"
                  onclick="window.back()"><br>
  window.forward() : <input type=button name="MyForward" value="前进"
                  onclick="window.forward()"><br>
  <p><b>NN4+和 IE3+公共方法 : </b></p>
  parent.Display.history.back() : <input type=button name="MyBack2" value="返回"
                  onclick="parent.Display.history.back()"><br>
  parent.Display.history.forward() : <input type=button name="MyForward2" value="前进"
                  onclick="parent.Display.history.forward()"><br>
  <p><b>访问当前框架历史记录 : </b></p>
  history.back() : <input type=button name="MyBack3" value="返回"
                  onclick="history.back()"><br>
  history.forward() : <input type=button name="MyForward3" value="前进"
                  onclick="history.forward()"><br>
  <p><b>访问父页面历史记录 : </b></p>
  parent.history.back() : <input type=button name="MyBack4" value="返回"
                  onclick="parent.history.back()"><br>
  parent.history.forward() : <input type=button name="MyForward4" value="前进"
                  onclick="parent.history.forward()"><br>
</form>
</center>
</body>
```

```
</html>
```

保存后运行“index.html”文档，显示如图 7.34 所示的页面。

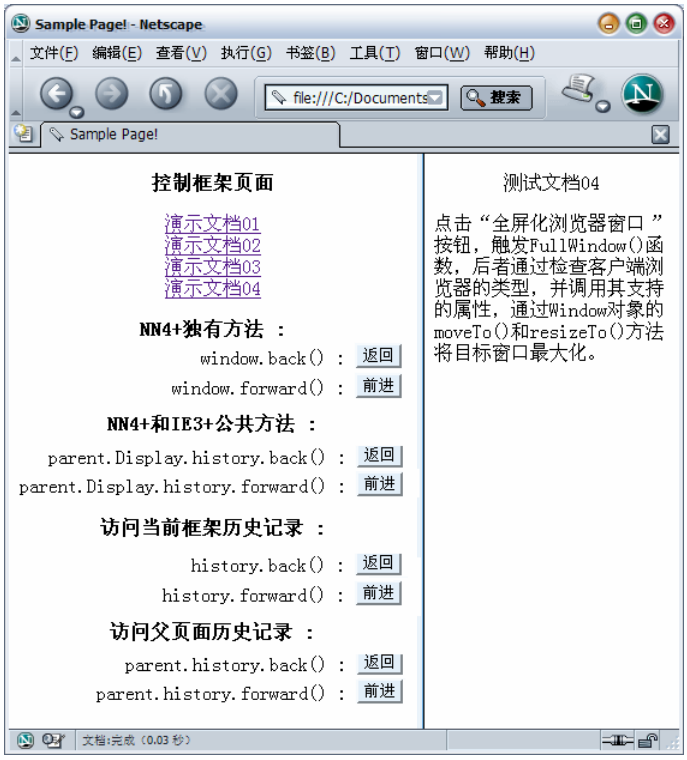

图 7.34 使用 back()和 forward()方法进行站点页面导航

除非要在导航到另外一地址之前需要进行一些附加操作，否则只需将 back()和 forward()方法作为参数传递给按钮定义的事件处理属性即可：

```
<input type=button name="MyButton" value="Forward" onclick="history.forward()">
```

出于兼容性考虑，一般通过框架对象的 parent 属性回溯到父文档中调用 back()和 forward()方法进行导航。

值得注意的是，History 对象的 back()和 forward()方法只能通过目标窗口或框架的历史 URL 地址记录列表分别向后和向前延伸，两者互为平衡。这两种方法有个显著的缺点，就是只能实现历史 URL 地址列表的顺序访问，而不能实现有选择的访问。为此，History 对象引入了 go()方法实现历史 URL 地址列表的随机访问。

7.5.2 使用 go()方法进行站点导航

History 对象提供另外一种站点导航的方法即 history.go(index|URLString)，该方法可接受两种形式的参数：

- 参数 index 传入导航目标页面与当前页面之间的相对位置，正整数值表示向前，负整数值表示向后。
- 参数 URLString 表示历史 URL 列表中目标页面的 URL，要使 history.go(URLString)方法有效，则 URLString 必须存在于历史 URL 列表中。

更改上节中的“left_main.html”文档，使用 history.go()方法进行站点页面导航：

```
//源程序 7.11
```

```
<!DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page!</title>
<script language="JavaScript" type="text/javascript">
<!--
function NavByNum()
{
   var num=document.MyForm.MyOffset.value;
   if(num>-10&&num<10)
      window.history.go(num);
   else
      alert("The input data is error!");
}
function NavByURL()
{
   var str=document.MyForm.MyURL.value;
   if(str)
      window.history.go(str);
   else
      alert("The input data is error!");
}
//-->
</script>
</head>
<body>
<center>
<p><b>控制框架页面</b></p>
<p>
   <a href="01.html" target="Display">演示文档 01</a><br>
   <a href="02.html" target="Display">演示文档 02</a><br>
   <a href="03.html" target="Display">演示文档 03</a><br>
   <a href="04.html" target="Display">演示文档 04</a><br>
   <a href="05.html" target="Display">演示文档 05</a><br>
   <a href="06.html" target="Display">演示文档 06</a><br>
   <a href="07.html" target="Display">演示文档 07</a><br>
   <a href="08.html" target="Display">演示文档 08</a><br>
   <a href="09.html" target="Display">演示文档 09</a><br>
   <a href="10.html" target="Display">演示文档 10</a><br>
</p>
<form name="MyForm">
   目标页面偏移 : <input type="text" name="MyOffset" id="MyOffset" value=1><br>
   <input type="button" name="MyButton1" value="使用偏移量导航"
   onclick="NavByNum()"><br>
   历史页面 URL : <input type="text" name="MyURL" id="MyURL"><br>
   <input type="button" name="MyButton2" value="使用历史页面 URL 导航"
   onclick="NavByURL()"><br>
</form>
</center>
</body>
</html>
```

该程序将文本框数值或URL地址作为参数调用History对象的go()方法实现站点页面的导航，其中“01.html”、“02.html”、…、“10.html”与源程序 7.9 中“01.html”文档结构类似，内容基本相同。主框架集文档“index.html”与源程序 7.8 相同。

运行“index.html”文档，显示如图 7.35 所示页面。

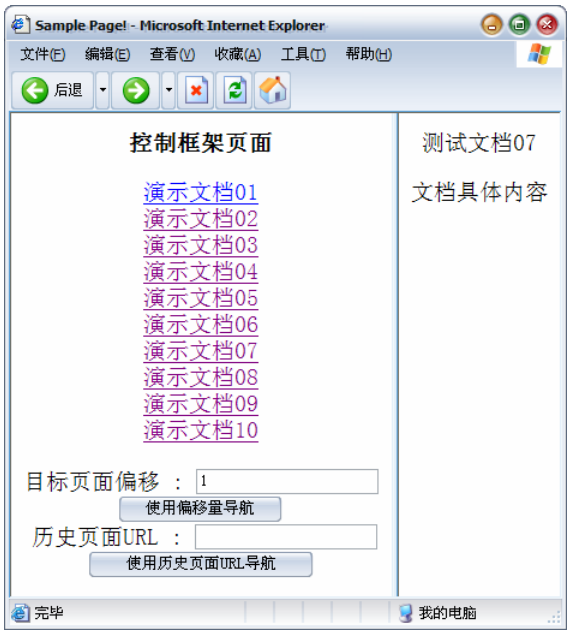

图 7.35 使用 go()方法进行站点页面导航

History 对象的 go()方法可传入参数 0 并设置合适的间隔时间计时器来实现文档页面重载。同时，history.go(-1)等同于 history.back()，history.go(1)等同于 history.forward()。

值得注意的是，go()方法在 IE 浏览器较老版本中获得的支持欠佳，具体表现在如下几个方面：

- IE3 中在 go()方法传入非 0 值，其结果相当于传入参数-1，即返回上一页面（如果存在的话）。同时，go()方法不接受字符串类型的历史 URL 地址；
- IE4 中，匹配字符串必须是 URL 的一部分，而不是文档标题的一部分。同时，传入参数 0 进行页面重载时，浏览器直接请求服务器返回重载页面，而不是从文档缓存中重载。

实际应用中，由于历史 URL 地址列表对用户而言一般为不可见的，所以其相对位置不确定，很难使用除-1、1 和 0 之外的参数调用 go()方法进行准确的站点页面导航。

7.5.3 常见属性和方法汇总

History 对象在处理历史 URL 地址列表并进行站点页面导航方面有着广泛的应用，表 7.5 列出了其常见的属性、方法以及浏览器支持情况。

表 7.5 History对象常见属性和方法汇总

类型	项目	简要说明	浏览器支持
属性	length	保存历史URL地址列表的长度信息	NN2+、IE3+
	current	在具有签名脚本的网页中指向历史URL列表的当前项	NN4+
	next	在具有签名脚本的网页中指向历史URL列表当前项的前一项	NN4+

	previous	在具有签名脚本的网页中指向历史URL列表当前项的下一项	NN4+
方法	back()	浏览器载入历史URL地址列表的当前URL的前一个URL	NN2+、IE3+
	forward()	浏览器载入历史URL地址列表的当前URL的下一个URL	NN2+、IE3+
	go(num|str)	浏览器载入历史URL列表中由参数num指定相对位置的URL地址对应的页面、或由参数str指定其URL地址对应的页面。	NN2+、IE3+

总的来说，JavaScript 脚本很难使用 History 对象提供的方法来管理历史 URL 地址列表或者明确当前 URL 在此列表中的相对位置，使得脚本在站点页面导航时精确定位相当困难，如每个方向能导航多远、指定偏移相对位置将跳转到哪个 URL 等。

理解了保存浏览器访问历史 URL 地址信息的 History 对象，下面介绍与浏览器当前文档 URL 信息相关的 Location 对象。

7.6 Location 对象

Location 对象在顶级对象模型中处于 Window 对象的下一个层次，用于保存浏览器当前打开的窗口或框架的 URL 信息。如果窗口含有框架集，则浏览器的 Location 对象保存其父窗口的 URL 信息，同时每个框架都有与之相关联的 URL 信息。在深入了解 Location 对象之前，先简单介绍 URL 的概念。

7.6.1 统一资源定位器（URL）

URL（Uniform Resource Locator：统一资源定位器，以下简称 URL）是 Internet 上用来描述信息资源的字符串，主要用在各种 WWW 客户程序和服务器程序上。采用 URL 可以用一种统一的格式来描述各种信息资源，包括文件、服务器地址和目录等。

URL 常见格式如下：

```
protocol://hostname[:port]/[path][?search][#hash]
```

参数的意义如下：

- protocol：指访问 Internet 资源和服务的网络协议。常见的协议有 Http、Ftp、File、Telnet、Gopher 等；
- hostname：指要访问的资源和服务所在的主机对应的域名，由 DNS 负责解析。例如 www.baidu.com、www.lenovo.com 等；
- port：指网络协议所使用的 TCP 端口号，此参数可选，并且在服务器端可自由设置。如 Http 协议常使用 80 端口等；
- path：指要访问的资源和服务相对于主机的路径，此参数可选。假设目标页面“query.cgi”相对于主机 hostname 的位置为/MyWeb/htdocs/，访问该页面的网络协议为 Http，则通过 http://hostname/MyWeb/htdocs/query.cgi 访问；
- search：指 URL 中传递的查询字符串，该字符串通过环境变量 QUERY_STRING 传递给 CGI 程序，并使用问号（?）与 CGI 程序相连，若有多项查询目标，则使用加号（+）连接，此参数可选。例如要在“query.cgi”中查询 name、number 和 code 信息，可通过语句 http://hostname/MyWeb/htdocs/query.cgi?name+number+code 实现；
- hash：表示指定的文件偏移量，包括散列号（#）和该文件偏移量相关的位置点名称，此参数可选。例如要创建与位置点“MyPart”相关联的文件部分的链接，可在链接的 URL 后添加“#MyPart”。

URL 是 Location 对象与目标文档之间联系的纽带。Location 对象提供的方法可通过传入的 URL 将文档装入浏览器，并通过其属性保存 URL 的各项信息，如网络协议、主机名、

端口号等。

> 注意：search 代表的产句字符串有多种形式，其形式与搜索引擎相关。如常见的名-值对应格式，用等号（=）分开名字与对应的值，多个名值之间使用连接符（&）连接。上述的搜索字符串可表示为：?name=zzangpu&num=200104&code=014104。在实际应用中常将查询字符串使用 escape()函数转换位 URL 适用的格式。

7.6.2 Location 对象属性与 URL 的对应

浏览器载入目标页面后，Location 对象的诸多属性保存了该页面 URL 的所有信息，其属性、方法及浏览器支持情况如表 7.6 所示。

表 7.6 Location对象的属性列表

类型	项目	简要说明	浏览器支持
属性	hash	保存URL的散列参数部分，将浏览器引导到文档中锚点	NN2+、IE3+
	host	保存URL的主机名和端口部分	NN2+、IE3+
	hostname	保存URL的主机名	NN2+、IE3+
	href	保存完整的URL	NN2+、IE3+
	pathname	保存URL完整的路径部分	NN2+、IE3+
	port	保存URL的端口部分	NN2+、IE3+
	protocol	保存URL的协议部分，包括协议之后的冒号	NN2+、IE3+
	search	保存URL的查询字符串部分	NN2+、IE3+
方法	assign(URL)	将以参数传入的URL赋予Location对象或其href属性	NN2+、IE3+
	reload(Boolean)	重载（刷新）当前页面	NN3+、IE4+
	replace(URL)	载入以参数URL传入的地址对应的文档	NN3+、IE4+

在 URL 载入后，其各个部分将分别由 Location 对象的各个属性保存起来。考察如下典型网页的 URL 地址实例：

```
http://www.webname.com:80/MyWeb/htdocs/query.cgi?name+num+code#MyPart3
```

浏览器载入该 URL 对应的页面时创建 Window 对象，并立即创建 Location 对象，且将 URL 的各个部分作为其属性值保存起来，如表 7.7 所示。

表 7.7 创建Location对象后各属性与其值的对应表

属性	取值
hash	#MyPart3
host	www.webname.com:80
hostname	www.webname.com
href	http://www.webname.com:80/MyWeb/htdocs/query.cgi?name+num+code#MyPart3
pathname	/MyWeb/htdocs/query.cgi
port	80
protocol	http:
search	?name+num+code

在使用 Location 对象的属性获得了 URL 地址的各个部分之后，可重新对属性赋值，实现页面跳转、锚点间转移、指定搜索串等功能。考察如下使用 Location 对象的 hash 属性进行锚点间跳转的实例代码：

```
//源程序 7.12
<!DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page!</title>
```

```
<script language="JavaScript" type="text/javascript">
<!--
var index=0;
var AnchorArray=new Array("StartAnchor","FirstAnchor","SecondAnchor",
                          "ThirdAnchor","ForthAnchor");
function GoAnchorNext()
{
   window.location.hash=AnchorArray[index];
   if(index<5)
   {
      index=index+1;
   }
   else
      index=0;
   return;
}
//-->
</script>
</head>
<body>
<center>
<p><b>锚点间跳转实例</b></p>
<p><a id="StartAnchor">Start Anchor</a></p>
<p><a id="FirstAnchor">First Anchor</a></p>
<p><a id="SecondAnchor">Second Anchor</a></p>
<p><a id="ThirdAnchor">Third Anchor</a></p>
<p><a id="ForthAnchor">Forth Anchor</a></p>
<form name="MyForm">
   <input type="button" name="MyButton" value="使用 hash 进行锚点转换"
   onclick="GoAnchorNext()"><br>
</form>
</center>
</body>
</html>
```

程序运行结果如图 7.36 所示。

图 7.36 使用 Location 对象的 hash 属性进行锚点间跳转

单击页面中的“使用 hash 进行锚点转换”按钮，锚点在五个锚点间循环移动。主要使用的语句为：

```
window.location.hash=AnchorArray[index];
```

同样，可以使用 Location 对象的 href 属性将浏览器页面导航到任意的 URL，方法如下：

```
window.location.href=newURL;
```

在框架集中，Location 对象的引用需遵循一定的原则，考察如下的包含框架集的简单网页代码：

```
//源程序 7.13
<html>
<head>
   <title>Sample Page!</title>
</head>
<frameset cols="50%,50%">
   <frame name="Frame01" src="01.html">
   <frame name="Frame02" src="02.html">
</frameset>
</html>
```

文档载入后，将产生几个 Location 对象，具体来说：

- window.location：显示运行包含此脚本的文档的框架 URL；
- parent.location：表示框架集的父窗口的 URL 信息；
- parent.frames[0].location：表示框架集中第一个可见框架的 URL 信息；
- parent.frames[1].location：表示框架集中第二个可见框架的 URL 信息；
- parent.otherFrameName.location：同一框架集中另一个框架的 URL 信息。

可见，对于框架集中某一个可见框架而言，访问其 URL 信息，需先将引用指向其父窗口，然后通过层次关系来调用。

7.6.3 使用 reload()方法重载页面

Location 对象的 reload()方法容易与浏览器工具栏中的 Reload/Refresh（重载/刷新）按钮发生混淆，但前者比后者可实现的重载方式要灵活得多。总体来讲，reload()方法可实现的重载方式主要有三种：

- 强制从服务器重载：每次刷新页面时，浏览器都默认请求 Web 服务器返回当前 URL 对应的页面给客户端，此法使用 true 作为参数；
- 启动会话时从服务器重载：如果 Web 服务器上的文档较之缓冲区内存放的文档新，或者缓冲区内根本没有这个文档，则从 Web 服务器重载当前 URL 对应的页面；
- 强制从缓冲区重载：每次刷新页面时，浏览器都默认请求浏览器从缓冲区载入当前 URL 对应的页面。如果缓冲区没有此页面，则从 Web 服务器重载。

前面已经讲述过，使用 History 对象的 go(0)方法也可实现页面的重载。从本质上讲，主要体现了两种不同性质的重载。

history.go(0)方法重载页面时，在缓冲区取得文档，并保持页面中许多对象的状态，仅改变其全局变量值及一些可设置但不可见的属性（如 hidden 输入对象的值等），该方法与浏览器工具栏内的 Reload/Refresh（重载/刷新）按钮功能相同。

location.reload(URL)方法重载页面时，不管是从 Web 服务器还是从缓冲区重载，目标页面内的所有对象都自动更新。

考察如下实现页面不同重载方式的代码：

```
//源程序 7.14
```

```
<!DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page!</title>
<script language="JavaScript" type="text/javascript">
<!--
function softReload()
{
   history.go(0);
}
function hardReload()
{
   location.reload(true);
}
//-->
</script>
</head>
<body>
<center>
<p>学籍注册程序</p>
<form name="MyForm">
   <p>姓名 : <input type="text" name="MyText" id="MyText" value="ZangPu"></p>
   <p>性别 : <input type="radio" name="MyRadio" id="MyRadio" value=1 checked>Male
              <input type="radio" name="MyRadio" id="MyRadio" value=2>Female</p>
   <p>年级 : <select name="MyClass" id="MyClass">
                 <option selected>Class 1</option>
                 <option>Class 2</option>
                 <option>Class 3</option>
                 <option>Class 4</option>
              </select>
   </p>
   <p><input type="button" name="Soft" value="使用 history.go(0)方法重载"
       onclick="softReload()"></p>
   <p><input type="button" name="Hard" value="使用 location.reload()方法重载"
       onclick="hardReload()"></p>
</form>
</center>
</body>
</html>
```

程序运行后，出现如图 7.37 所示的页面。

图 7.37 程序运行后的原始页面

在原始页面表单中更改“姓名”栏为“YSQ”、“性别”栏为“Female”、“年级”栏为“Class 3”后，出现如图 7.38 所示页面。

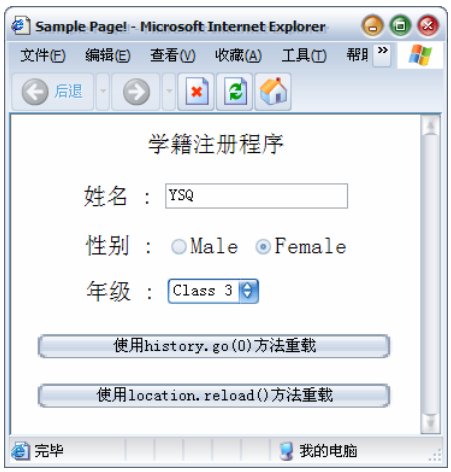

图 7.38 更改表单中各项参数

单击“使用 history.go(0)方法重载”按钮后，页面刷新但表单内容不变，结果如图 7.38 所示；单击“使用 location.reload()方法重载”按钮后，页面刷新且表单内容发生变化，结果如图 7.37 所示。

Location 对象在搜索引擎和页面的重定位、重载等方面应用比较广泛，实际使用过程中应综合使用字符串处理的有关方法如 substring()、escape()和 unescape()等方法分析目标 URL 地址以得到有用的信息。下面介绍与 Window 对象紧密相连的 Frame 对象。

7.7 Frame 对象

框架在 Web 应用程序设计中存在着很大的争议，某些场合使用框架能简化设计步骤，如多页面导航实例中，可在一个固定的框架内添加导航控件如图片、按钮等，而在另外的框架中显示导航的结果，这样客户端随时都可通过该导航控件控制其它框架的显示内容而不需

刷新整个文档。

浏览器载入含有框架的文档时自动创建 Frame 对象，并允许脚本通过调用 Frame 对象的属性和方法来控制页面中的框架。在 IE 和 NN 浏览器中，一般将 Frame 对象实现为 Window 对象，但前者不支持 close()方法关闭框架自身，并且 Frame 对象拥有其独有的属性和方法用于框架操作。在介绍 Frame 对象之前，先来了解框架集文档中对象的结构层次。

7.7.1 框架集文档中对象的结构

对于单个、无框架的文档，对象模型以 Window 对象开始，并将其作为对象模型层次的起始点和默认的对象而存在，实际使用过程中可忽略对该对象的引用。

对于含有多个框架的框架集文档，该框架集文档作为父窗口，且此时浏览器的标题栏显示的是框架集文档的标题。考察如下最简单的框架集文档代码：

```
//源程序 7.15
<html>
<head>
   <title>Sample Page!</title>
</head>
<frameset cols="50%,50%">
   <frame name="Frame01" id="Frame01" src="01.html">
   <frame name="Frame02" id="Frame02" src="02.html">
</frameset>
</html>
```

每个<Frame>标记都可以加载一个新的文档（当然也可引入新的框架集文档形成更为复杂的框架集文档）而产生各自的 document 对象，并可通过对象之间的层次关系进行访问操作。父子对象之间通过 parent 对象进行联系，且存在 top 对象指向所有框架唯一共有的顶层窗口。图 7.39 显示了上述框架集文档的结构：

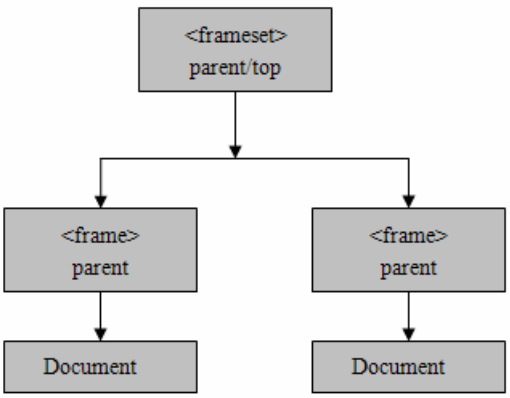

图 7.39 简单的框架集文档模型

在对象模型中构造一个对象的引用，首先要获得目标对象的位置信息，然后应了解用何种方法来实现该引用。在框架集文档中对框架的引用路径主要有三种：

- 子到父
- 父到子
- 子对子

在子到父的引用路径中，可使用 parent 关键字实现；父到子的引用可直接使用对象模型

层次；而子到子的访问则需要通过 top 关键字引用其共有的父对象，然后通过该父对象实现对另一框架的访问。综合如下：

```
this.parent;
parent.frameName;
top.otherFrameName.document;
```

Frame 对象的属性和方法受<frame>标记的控制，可在此标记内设定该框架的相关信息，如框架是否有滚动条、边框的颜色等。一般而言，在<frame>标记内应设置其 ID 属性（或 name 属性）以实现对象的有效引用。在上述最简单的框架集文档中，可通过如下方法实现对框架 Frame02 的 frameBorder 属性的访问（假设操作焦点在 Frame01 框架中）：

```
parent.document.all.Frame02.frameBorder;
parent.document.getElementById("Frame02").frameBorder;
```

在嵌套的框架集文档中，可根据对象模型层次从顶层的 Window 对象开始引用并到达最终的 Frame 对象，然后通过其属性和方法来操作该框架中载入的文档。

> 注意：如果框架集中某个框架载入了另外的框架集文档，则该框架的 top 属性属于另一个框架集文档中定义的框架集。如果总是把 top 设定为父对象，则该引用将不可实现。实际应用中可通过判断顶级文档是否为其 top 或 parent 属性载入，并根据结果实施页面重定位的方法来禁止框架中载入新的框架集文档，进而解决对象引用失效的问题。

7.7.2 控制指定的框架

Frame 对象提供诸多的属性和方法用于控制制定的框架，如更改框架是否能改变大小的标志、是否显示框架的滚动条等。

NN6+浏览器中实现的 Frame 对象提供 contentDocument 属性来引用与当前框架载入的文档相关的 document 对象，从而获取框架中其它有用的信息。IE5.5+和 NN7 浏览器实现的 Frame 对象提供 contentWindow 属性来引用与当前框架产生的窗口相关的 Window 对象，从而根据文档对象模型来获取 document 对象及其他信息。

例如某框架集文档包含两个框架分别为“FrameName1”和“FrameName2”，而 targetWin 是与“FrameName2”框架相关的 Window 对象，则在“FrameName1”框架所载入的文档中可通过下面句子实现对 targetWin 的引用：

```
var targetWin=parent.document.getElementById("FrameName1").contentWindow;
```

考察如下使用 Frame 对象的属性和方法控制框架的实例。首先考虑包含左右两个框架并设定了相关参数的框架集文档：

```
//源程序 7.16
<html>
<head>
   <title>Sample Page!</title>
</head>
<frameset name="MyFrameset" border=3 borderColor="black" cols="60%,40%">
   <frame name="Control" src="leftmain.html">
   <frame name="Display" frameBorder="yes" borderColor="#c0c0c0" src="target.html">
</frameset>
</html>
```

其中右侧框架中载入的“target.html”文档为“学籍注册程序”页面，包含文本框、单选框和下拉框等。其程序代码如下：

```
//源程序 7.17
<!DOCTYPE HTML PUBLIC   "-//W3C//DTD HTML 4.0//EN"
```

```
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page!</title>
</head>
<body>
<center>
<p>学籍注册程序</p>
<form name="MyForm">
  <p>姓名 : <input type="text" name="MyName" id="MyName" value="ZangPu"></p>
  <p>性别 : <input type="radio" name="MySex" id="MySex" value="男" checked>男
            <input type="radio" name="MySex" id="MySex" value="女">女</p>
  <p>年级 : <select name="MyClass" id="MyClass">
              <option value="class 1" selected>Class 1</option>
              <option value="class 2" >Class 2</option>
              <option value="class 3" >Class 3</option>
              <option value="class 4" >Class 4</option>
            </select>
  </p>
  <p>
  <input type="button" name="MySubmit" value="确认"
      onclick="MySubmitFunc()">
  <input type="button" name="MyCancle" value="取消"
      onclick="MyCancleFunc()">
  </p>
</form>
</center>
</body>
</html>
```

其中 MySubmitFunc()、MyCancleFunc()事件处理程序为演示程序，读者可自行扩充以实现更为复杂的功能。框架集左侧框架载入的“leftmain.html”文档的代码如下：

```
//源程序 7.18
<!DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page!</title>
<script language="JavaScript" type="text/javascript">
<!--
//设置右侧框架信息
function SetInfo()
{
  //通过右侧框架对象的 name 属性定位该框架
  var targetFrame=parent.document.getElementById(parent.document.frames[1].name);
  //获取左侧框架中的"框架名称"文本框对象，并修改右侧框架对象的 name 属性
  var leftFrameName=parent.document.frames[0].document.all.MyForm.FrameName;
  if(leftFrameName.value=="")
  {
    alert("右侧框架名称信息不能为空!");
    leftFrameName.focus();
  }
```

```
    else
    {
        targetFrame.name=leftFrameName.value;
    }
    //获取左侧框架中的"边框显示"单选框对象,并设置右侧框架对象的 frameBorder 属性
    var leftFrameRadio=parent.document.frames[0].document.all.MyForm.MyRadio;
    for(i=0;i<leftFrameRadio.length;i++)
    {
        if(leftFrameRadio[i].checked)
            targetFrame.frameBorder=leftFrameRadio[i].value;
    }
    //获取左侧框架中的"边框颜色"下拉框对象,并设置右侧框架对象的 borderColor 属性
    var leftFrameColor=parent.document.frames[0].document.all.MyForm.MyColor;
    for(i=0;i<leftFrameColor.length;i++)
    {
        if(leftFrameColor[i].selected)
            targetFrame.borderColor=leftFrameColor[i].value;
    }
    //获取左侧框架中"姓名字段"文本框对象的 value 值,并修改右侧框架中"姓名"字段 value 属性
    var iMyName=parent.document.frames[0].document.all.MyForm.MyNameLeft.value;
    parent.document.frames[1].document.all.MyForm.MyName.value=iMyName;
    //获取左侧框架中"性别字段"的选择结果,并修改右侧框架中"性别"字段选择结果
    var iCheck=parent.document.frames[0].document.all.MyForm.MySexLeft;
    for(i=0;i<iCheck.length;i++)
    {
        if(iCheck[i].checked)
            parent.document.frames[1].document.all.MyForm.MySex[i].checked=true;
    }
    //获取左侧框架中"年级字段"的选择结果,并修改右侧框架中"年级"字段选择结果
    var iSelect=
    parent.document.frames[0].document.all.MyForm.MyClassLeft.options.selectedIndex;
    parent.document.frames[1].document.all.MyForm.MyClass.selectedIndex=iSelect;
}
//获取右侧框架信息
function AlertInfo()
{
    var nameTemp;
    var sexTemp;
    var classTemp;
    //获取右侧框架对象
    var targetFrame=parent.document.getElementById(parent.document.frames[1].name);
    //获取右侧框架中"姓名"文本框
    var iName=parent.document.frames[1].document.all.MyForm.MyName;
    nameTemp=iName.value;
    //获取右侧框架中"性别"单选框,并确定选中状态
    var iRadio=parent.document.frames[1].document.all.MyForm.MySex;
    for(i=0;i<iRadio.length;i++)
    {
        if(iRadio[i].checked)
            sexTemp=iRadio[i].value;
    }
    //获取右侧框架中"年级"下拉框,并确定选中状态
    var iClass=parent.document.frames[1].document.all.MyForm.MyClass;
```

```
  for(i=0;i<iClass.length;i++)
  {
     if(iClass[i].selected)
        classTemp=iClass[i].value;
  }
  //输出相关信息
  var msg="\n 右侧框架信息 :                                                    \n\n";
  msg+="          框架名称 : "+targetFrame.name+"\n";
  msg+="          边框显示 : "+targetFrame.frameBorder+"\n";
  msg+="          边框颜色 : "+targetFrame.borderColor+"\n";
  msg+="          姓名字段 : "+nameTemp+"\n";
  msg+="          性别字段 : "+sexTemp+"\n";
  msg+="          年级字段 : "+classTemp+"\n";
  alert(msg);
}
//-->
</script>
</head>
<body>
<center>
<p>设置右侧框架信息</p>
<form name="MyForm">
  <p>框架名称 : <input type="text" name="FrameName" id="FrameName" value="Right"></p>
  <p>边框显示 : <input type="radio" name="MyRadio" id="MyRadio" value="yes" checked>是
                <input type="radio" name="MyRadio" id="MyRadio" value="no">否</p>
  <p>边框颜色 : <select name="MyColor" id="MyColor">
                   <option value="black" selected>黑色</option>
                   <option value="red">红色</option>
                   <option value="blue">蓝色</option>
                   <option value="green">绿色</option>
                </select>
  </p>
  //设置右侧框架的元素对象信息
  <p>姓名字段 : <input type="text" name="MyNameLeft" id="MyNameLeft" value="ZangPu"></p>
  <p>性别字段 : <input type="radio" name="MySexLeft" id="MySexLeft" value="男" checked>男
                <input type="radio" name="MySexLeft" id="MySexLeft" value="女">女</p>
  <p>年级字段 : <select name="MyClassLeft" id="MyClassLeft">
                   <option value="Class 1" selected>Class 1</option>
                   <option value="Class 2" >Class 2</option>
                   <option value="Class 3" >Class 3</option>
                   <option value="Class 4" >Class 4</option>
                </select>
  <p>
  <p><input type="button" name="setInfo" value="提交设置" onclick="SetInfo()"></p>
</form>
<hr>
<p>显示右侧框架信息</p>
<form>
  <p><input type="button" name="alertInfo" value="获取信息" onclick="AlertInfo()"></p>
</form>
</center>
</body>
</html>
```

程序运行后，出现如图 7.40 所示的页面。

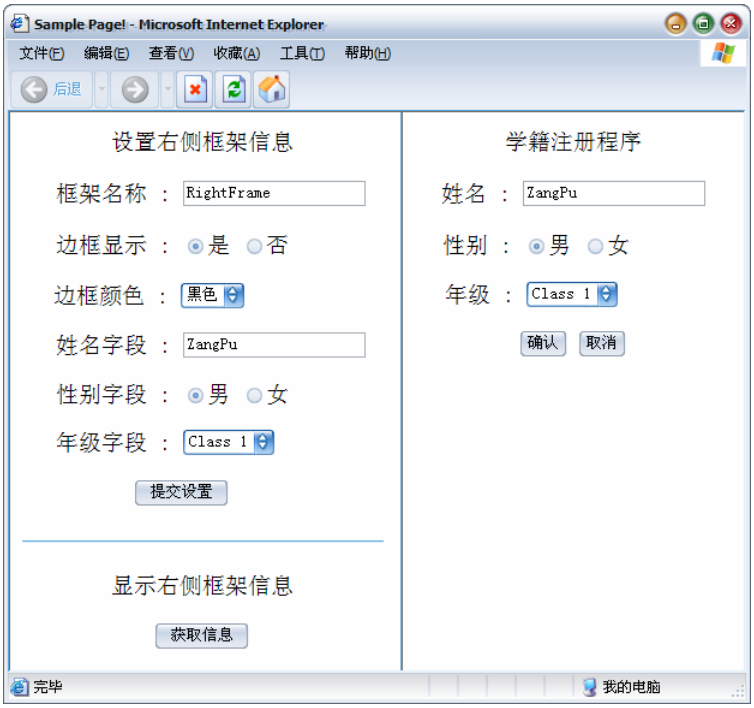

图 7.40 程序运行后的原始页面

此时单击“获取信息”按钮，则弹出包含右侧框架及其对应得 Frame 对象相关信息的警告框，如图 7.41 所示。

图 7.41 页面载入时根据页面初始值设定的右侧框架信息

在原始页面更改右侧框架及 Frame 对象的相关信息，将 Frame 对象的 name 属性改为“RightFrame”，frameBorder 属性改为“no”，borderColor 属性改为“red”，“姓名字段”文本框改为“YSQ”，“性别字段”单选框改为“女”，“年级字段”下拉框改为“class 3”。然后单击“提交设置”按钮后更新右侧框架及其对应的 Frame 对象相关信息。

单击“获取信息”按钮，弹出包含右侧框架及对应的 Frame 对象相关信息的警告框，如图 7.42 所示。

图 7.42 更改相关设置后更新右侧框架及 Frame 对象相关信息

不同的浏览器通常都会在文档内容和框架之间添加空白区域以便文档载入时其内容能自动插入到框架中。Frame 对象提供属性 marginHeight(上边界和下边界)、marginWidth(左边界和右边界)来表示该空白区域的大小。

值得注意的是，Frame 对象的属性如 frameBorder、marginHeight 等都可读可写，但框架集加载后，改变这些属性的取值能够改变其具体内容，并不能改变框架中文档的外观，而需通过 document 对象调用其对应的属性和方法来修改。

7.7.3 常见属性和方法汇总

Frame 对象提供的属性和方法较少，主要用于设置框架的标记（如 name、src 等属性）、改变框架的外观（如 borderColor、scrolling 等属性）等，表 7.8 列出了其常见的属性及浏览器版本支持情况。

表 7.8 Frame对象常见的属性汇总

属性	简要说明	浏览器支持
allowTransparency	标记框架的背景是否透明，为Boolean值	IE6+
borderColor	设置框架边框的颜色，为三组16进制值或颜色字符串	IE4+
contentDocument	对框架载入的目标文档对应的document对象的引用	NN6+
contentWindow	对与框架对应的窗口相关的Window对象的引用	NN7+、IE5.5+
frameBorder	设置框架是否包含边框，为字符串“yes”或“no”	NN6+、IE4+
height	保存框架的高度信息（像素）	IE4+
marginHeight	保存框架与所载入文档之间空白区域的高度（像素）	NN6+、IE6+
marginWidth	保存框架与所载入文档之间空白区域的宽度（像素）	NN6+、IE6+
name	返回与框架相关的标识符，用于框架的引用	NN6+、IE4+
noResize	表示框架是否能改变大小的状态标识符	NN6+、IE6+
scrolling	表示框架打开和关闭滚动条的状态标识符	NN6+、IE6+
src	框架载入的文档对应的URL地址	NN6+、IE6+
width	保存框架的宽度信息（像素）	IE4+

注意：如果某属性的取值为 Boolean 类型，则在调用该属性时可使用 1 代替 true、0 代替 false 作为其取值或返回的结果，下同。

框架（Frame）一般包含在框架集（Frameset）中，并且一个框架集一般包含多个框架。下面简要介绍与框架集相关的 Frameset 对象和 iframe 元素对象。

7.7.4 Frameset 对象

Frameset 对象主要用于处理框架与框架之间的关系，如框架之间边框的厚度（像素）、颜色及框架集的大小等。浏览器载入包含框架集的文档时，生成 Frameset 对象，表 7.9 列出了其常见的属性及浏览器版本支持情况。

表 7.9 Frameset对象常见属性

属性	简要说明	浏览器支持
border	保存框架集中各框架之间的边框厚度信息（像素）	IE4+
borderColor	保存框架边框的颜色，为三组16进制值或颜色字符串	IE4+
cols	保存框架集cols信息的字符串，以百分比或星号引入	NN6+、IE4+
frameBorder	设置框架是否包含边框，为字符串“yes”或“no”	IE4+
frameSpacing	保存框架集中各框架之间的间距（像素）	IE4+
rows	保存框架集rows信息的字符串，以百分比或星号引入	NN6+、IE4+

在框架集文档中，如果某属性未被指定，则该属性为空，浏览器一般以该属性的默认值来定义框架集，如框架集文档定义中 frameSpacing 属性未被指定时，该框架集中框架之间的间距为默认值 2 像素。

一般而言，在框架集中访问 Frameset 对象的属性可通过如下方法：

```
（IE4+）          document.all.FramesetID.property
（IE5+/W3C）      document.getElementById(FramesetID).property
```

在框架集的某个框架内，可通过如下方法访问该 Frameset 对象：

```
（IE4+）          parent.document.all.FramesetID.property
（IE5+/W3C）      parent.document.getElementById(FramesetID).property
```

考察如下的框架集文档代码：

```
//源程序 7.19
<!DOCTYPE HTML PUBLIC   "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
  <title>Sample Page!</title>
</head>
<frameset name="FramesetOuter" id="FramesetOuter" border=5 frameBorder="yes"
            borderColor="green" frameSpacing=10 rows="290,*" cols="60%,100,100">
  <frame name="Control"   scrolling="no" src="main.html">
  <frame name="Control"   scrolling="no" src="target1.html">
  <frame name="Control"   scrolling="no" src="target2.html">
  <frame name="Control"   scrolling="no" src="target3.html">
  <frame name="Control"   scrolling="no" src="target4.html">
  <frame name="Control"   scrolling="no" src="target5.html">
</frameset>
</html>
```

该框架集包含六个框架，其中“target1.html”、“target2.html”、…、“target5.html”文档格式基本相同，文档“target1.html”的代码如下：

```
//源程序 7.20
<!DOCTYPE HTML PUBLIC   "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page!</title>
```

```
</head>
<body>
<center>
<br>
<p>学籍注册</p>
步骤之一
</center>
</body>
</html>
```

框架一载入的文档“main.html”实现对 Frameset 对象各属性的访问，同时可根据文档中文本框、单选框和下拉框对应的值更新该对象的各项属性，同时更新框架集文档视图。其代码如下：

```
//源程序 7.21
<!DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page!</title>
<script language="JavaScript" type="text/javascript">
<!--
var msg="\n 框架集信息 : \n\n";
msg+="结果(修改前) : \n";
//修改指定框架集各项属性并输出相关信息
function SetInfo()
{
  AlertInfo();
  var iFrameset=parent.document.getElementById("FramesetOuter");
  //设置边框显示与否
  var iFrameBorder=parent.document.frames[0].document.all.MyForm.MyFrameBorder;
  for(i=0;i<iFrameBorder.length;i++)
  {
    if(iFrameBorder[i].checked)
      iFrameset.frameBorder=iFrameBorder[i].value;
  }
  //设置边框厚度
  var iBorder=parent.document.frames[0].document.all.MyForm.MyBorder.value;
  if(iBorder==""||parseInt(iBorder)==NaN||parseInt(iBorder)<=0)
  {
    alert("\n 错误提示信息 : \n\n"边框厚度"字段所填数据不合法!\n");
  }
  else
    iFrameset.border=iBorder;
  //设置边框颜色
  var iBorderColor=
  parent.document.frames[0].document.all.MyForm.MyBorderColor.options.selectedIndex;
  iFrameset.BorderColor=
  parent.document.frames[0].document.all.MyForm.MyBorderColor.options[iBorderColor].value;
  //设置框架间距
  var iFrameSpacing=parent.document.frames[0].document.all.MyForm.MyFrameSpacing.value;
  if(iFrameSpacing==""||parseInt(iFrameSpacing)==NaN||parseInt(iFrameSpacing)<=0)
  {
    alert("\n 错误提示信息 : \n\n"框架间距"字段所填数据不合法!\n");
```

```
    }
    else
       iFrameset.frameSpacing=iFrameSpacing;
    //设置行高信息
    iFrameset.rows=parent.document.frames[0].document.all.MyForm.MyRows.value;
    iFrameset.cols=parent.document.frames[0].document.all.MyForm.MyCols.value;
    msg+="结果(修改后) : \n";
    AlertInfo();
    alert(msg);
    return;
}
//获取指定框架集信息
function AlertInfo()
{
    var iFrameset=parent.document.getElementById("FramesetOuter");
    msg+="          边框显示 : " +iFrameset.frameBorder+ "\n";
    msg+="          边框厚度 : " +iFrameset.border+ "\n";
    msg+="          边框颜色 : " +iFrameset.borderColor+ "\n";
    msg+="          框架间距 : " +iFrameset.frameSpacing+ "\n";
    msg+="          行高信息 : " +iFrameset.rows+ "\n";
    msg+="          列宽信息 : " +iFrameset.cols+ "                    \n\n";
}
//-->
</script>
</head>
<body>
<center>
<br>
设置并显示框架集信息
<form name="MyForm">
    边框显示 : <input type="radio" name="MyFrameBorder" id="MyFrameBorder" value="yes"
               checked>是
               <input type="radio" name="MyFrameBorder" id="MyFrameBorder" value="no">否
               <br>
    边框厚度 : <input type="text" name="MyBorder" id="MyBorder" size="11" value="5"><br>
    边框颜色 : <select name="MyBorderColor" id="MyBorderColor">
                   <option value="black" selected>黑色</option>
                   <option value="red">红色</option>
                   <option value="blue">蓝色</option>
                   <option value="green">绿色</option>
               </select><br>
    框架间距 : <input type="text" name="MyFrameSpacing" id="MyFrameSpacing" size="11"
               value="10"><br>
    行高信息 : <input type="text" name="MyRows" id="MyRows" size="11" value="290,*"><br>
    列宽信息 : <input type="text" name="MyCols" id="MyCols" size="11" value="60%,100,100"><br>
    <p><input type="button" name="setInfo" value="提交更改并获取信息" onclick="SetInfo()"></p>
</form>
</center>
</body>
</html>
```

程序运行后，出现如图 7.43 所示的页面。

图 7.43 程序运行后的原始页面

在该页面中修改表单中各项 HTML 元素对应的值，如选择“边框显示”为“否”、“框架间距”为 5，单击“提交更改并获取信息”按钮，将触发 SetInfo()函数更新 Frameset 对象各个对应的属性。浏览器根据 Frameset 对象的新属性值来刷新目标页面的视图，如图 7.44 所示。

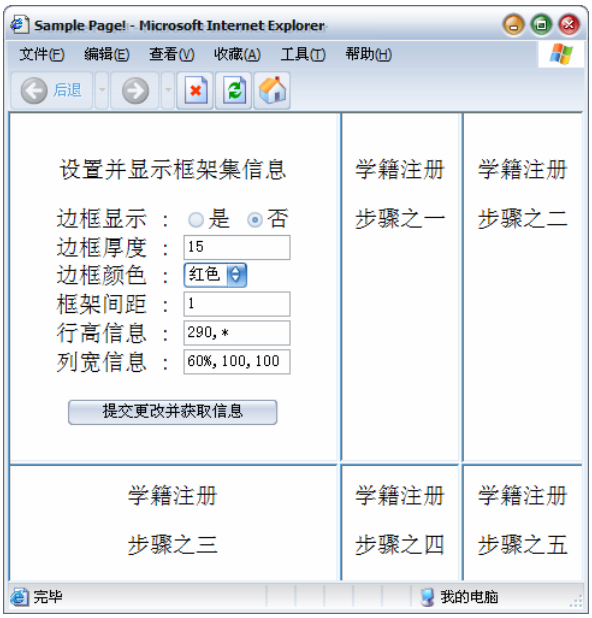

图 7.44 更改 Frameset 对象的属性后刷新视图

在更新目标页面视图的同时，将弹出包含修改前后框架集文档对应的 Frameset 对象属性值对比信息的警告框。如选择“边框显示”为“否”、“边框厚度”为 10、“框架间距”为 20，并单击“提交更改并获取信息”按钮后，浏览器更新目标页面视图并弹出警告框如图 7.45 所示。

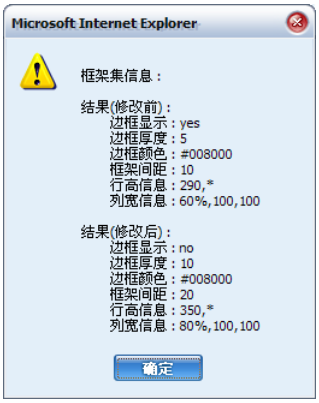

图 7.45 修改前后 Frameset 对象各属性值对比

值得注意的是，Frameset 对象的属性均为可读可写，但在框架集载入后，某些属性的更改并不反映到页面布局上来，如 frameBorder、border 属性等，其更多的是作为可读的属性而存在。

由上面的实例不难看出，框架集文档中的 Frameset 对象在框架管理方面提供快捷的途径来操作指定的项目并能实时更新目标页面。当框架集文档中所含的框架比较多或结构嵌套比较复杂的情况下，一般设定框架的 name 属性（或 id 属性），然后通过父窗口 document 对象的 getElementById(FrameID)方法准确定位，并进行相关操作。

7.7.5 iframe 元素对象

iframe 元素对象本质上是通过<iframe>和</iframe>标记对嵌入目标文档到父文档时所产生的对象，表示浮动在父窗口中的目标文档，表 7.10 列出了其常见的属性及浏览器版本支持情况。

表 7.10 iframe元素对象常见属性汇总

属性	简要说明	浏览器支持
align	根据文档中其他元素的位置对目标iframe元素进行定位	NN6+、IE4+
allowTransparency	标识iframe的背景是否透明	IE6+
contentDocument	包含对iframe框架内文档对象的引用	NN6+
contentWindow	包含对iframe框架所产生的Window对象的引用	NN7+、IE5.5+
height	保存iframe框架的高度信息（像素）	NN6+、IE4+
Hspace	保存iframe框架左右边框的页边空白（像素）	IE4+
longDesc	提供一个包含iframe元素详细描述的文档的URL	NN6+、IE6+
marginHeight	保存iframe框架上下边框与外部元素之间的间隔（像素）	NN6+、IE4+
marginWidth	保存iframe框架左右边框与外部元素之间的间隔（像素）	NN6+、IE4+
name	标识目标iframe框架的字符串	NN6+、IE4+
scrolling	标识目标iframe框架是否含有的滚动条的Boolean值	NN6+、IE4+
src	保存嵌入目标iframe框架内文档的URL	NN6+、IE4+
Vspace	保存iframe框架上下边框的页边空白（像素）	IE4+
Width	保存iframe框架的宽度信息（像素）	NN6+、IE4+

iframe 元素对象的访问方法与 Frameset 对象类似，在父文档中访问 iframe 元素对象的属性可通过如下方式：

```
（IE4+）          document.all.iframeID.property
（IE4+/NN6+）     window.frames[iframeID].property
（IE5+/W3C）      document.getElementById(iframeID).property
```

在包含 iframe 元素对象的文档内，可通过如下方式访问该 iframe 元素对象：

（IE4+）　　parent.document.all.iramesetID.property
（IE5+/W3C）　parent.document.getElementById(FramesetID).property

考察如下通过 iframe 元素对象的属性操作其对应框架的代码：

```
//源程序 7.22
<!DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page!</title>
<script language="JavaScript" type="text/javascript">
<!--
var msg="\niframe 元素对象对应的框架信息 : \n\n";
msg+="结果(修改前) : \n";
//返回下拉框被选择项的序号
function CheckIndex(iSelect)
{
   var index;
   for(i=0;i<iSelect.length;i++)
   {
      if(iSelect[i].selected)
         index=i;
   }
   return index;
}
//修改指定框架集信息
function SetInfo()
{
   ChangeInfo();
   msg+="结果(修改后) : \n";
   var iFrame=document.getElementById("MyIframe");
   //对齐方式
   var iAlign=document.all.MyForm.MyAlign;
   iFrame.align=iAlign[CheckIndex(iAlign)].value;
   //背景透明
   var iAllowTransparency=document.all.MyForm.MyAllowTransparency;
   iFrame.allowTransparency=iAllowTransparency[CheckIndex(iAllowTransparency)].value;
   //框架高度
   var iHeight=document.all.MyForm.MyHeight;
   iFrame.height=iHeight[CheckIndex(iHeight)].value;
   //框架宽度
   var iWidth=document.all.MyForm.MyWidth;
   iFrame.width=iWidth[CheckIndex(iWidth)].value;
   //空白区域高度
   var iMarginHeight=document.all.MyForm.MyMarginHeight;
   iFrame.marginHeight=iMarginHeight[CheckIndex(iMarginHeight)].value;
   //空白区域宽度
   var iMarginWidth=document.all.MyForm.MyMarginWidth;
   iFrame.marginWidth=iMarginWidth[CheckIndex(iMarginWidth)].value;
   //滚动显示
   var iScrolling=document.all.MyForm.MyScrolling;
   iFrame.scrolling=iScrolling[CheckIndex(iScrolling)].value;
   //文档地址
```

```
  var iSrc=document.all.MyForm.MySrc;
  if(iSrc.value=="")
  {
    alert(""文档地址"字段不能为空!");
    iSrc.focus();
  }
  else
  {
    iFrame.src=iSrc.value;
  }
  ChangeInfo();
  //输出信息并更新页面
  alert(msg);
  return;
}
//获取框架信息
function ChangeInfo()
{
  var iFrame=document.getElementById("MyIframe");
  msg+="          框架对齐 : " +iFrame.align+ "\n";
  msg+="          背景透明 : " +iFrame.allowTransparency+ "\n";
  msg+="          框架高度 : " +iFrame.height+ "\n";
  msg+="          框架宽度 : " +iFrame.width+ "\n";
  msg+="          空白高度 : " +iFrame.marginHeight+ "\n";
  msg+="          空白宽度 : " +iFrame.marginWidth+ "\n";
  msg+="          水平空白 : " +iFrame.hspace+ "\n";
  msg+="          垂直空白 : " +iFrame.vspace+ "\n";
  msg+="          滚动显示 : " +iFrame.scrolling+ "\n";
  msg+="          文档地址 : " +iFrame.src+ "\n\n";
  return;
}
//-->
</script>
</head>
<body>
<table height=380 width=650 border=2 borderColor="#c0c0c0">
  <tr>
    <td height=380 width=250>
      <center>设置并显示框架集信息</center>
      <form name="MyForm">
        框架对齐 : <select name="MyAlign" id="MyAlign">
                    <option value="absbottom">absbottom</option>
                    <option value="absmiddle">absmiddle</option>
                    <option value="baseline">baseline</option>
                    <option value="bottom">bottom</option>
                    <option value="left">left</option>
                    <option value="middle" selected>middle</option>
                    <option value="right">right</option>
                    <option value="texttop">texttop</option>
                    <option value="top">top</option>
                  </select><br>
        背景透明 : <select name="MyAllowTransparency" id="MyAllowTransparency">
                    <option value="true" selected>true</option>
```

```
                    <option value="false">false</option>
                  </select><br>
        框架高度 : <select name="MyHeight" id="MyHeight">
                    <option value=200 selected>200pixels</option>
                    <option value=250>250pixels</option>
                    <option value=300>300pixels</option>
                    <option value=350>350pixels</option>
                  </select><br>
        框架宽度 : <select name="MyWidth" id="MyWidth">
                    <option value=200 selected>200pixels</option>
                    <option value=250>250pixels</option>
                    <option value=300>300pixels</option>
                    <option value=350>350pixels</option>
                  </select><br>
        空白高度 : <select name="MyMarginHeight" id="MyMarginHeight">
                    <option value=10 selected>10 pixels</option>
                    <option value=15 selected>15 pixels</option>
                    <option value=20>20 pixels</option>
                    <option value=25>25 pixels</option>
                  </select><br>
        空白宽度 : <select name="MyMarginWidth" id="MyMarginWidth">
                    <option value=10>10 pixels</option>
                    <option value=15 selected>15 pixels</option>
                    <option value=20>20 pixels</option>
                    <option value=25>25 pixels</option>
                  </select><br>
        滚动显示 : <select name="MyScrolling" id="MyScrolling">
                    <option value="auto">auto</option>
                    <option value="yes" selected>yes</option>
                    <option value="no">no</option>
                  </select><br>
        文档地址 : <input type="text" name="MySrc" id="MySrc" size="8" value="other.html"><br>
        <p><input type="button" name="setinfobutton" value="提交更改并获取信息"
             onclick="SetInfo()"></p>
      </form>
    </td>
    <td height=380 width=400>
        框架集对象的属性均为可读可写,但某些属性的更改并不反映到页面视图中.
        <iframe name="MyIframe" id="MyIframe" align="absbottom" allowTransparency=true
                height=200 width=200 frameborder=10 marginHeight=15 marginWidth=15
                scrolling="yes"   hspace=20 vspace=10 src="target.html">
        </iframe>
        当框架集文档中所含的框架比较多时,一般需设定框架的 name 属性或 id 属性.
    </td>
  </tr>
</table>
</body>
</html>
```

其中“空白高度”和“空白宽度”表示框架中的内容与边框之间的距离（像素），而“水平空白”和“垂直空白”表示框架与其外部文档之间的距离（像素）；测试文档“target.html”和“other.html”为普通的 HTML 页面。

程序运行后，出现如图 7.46 所示的页面。

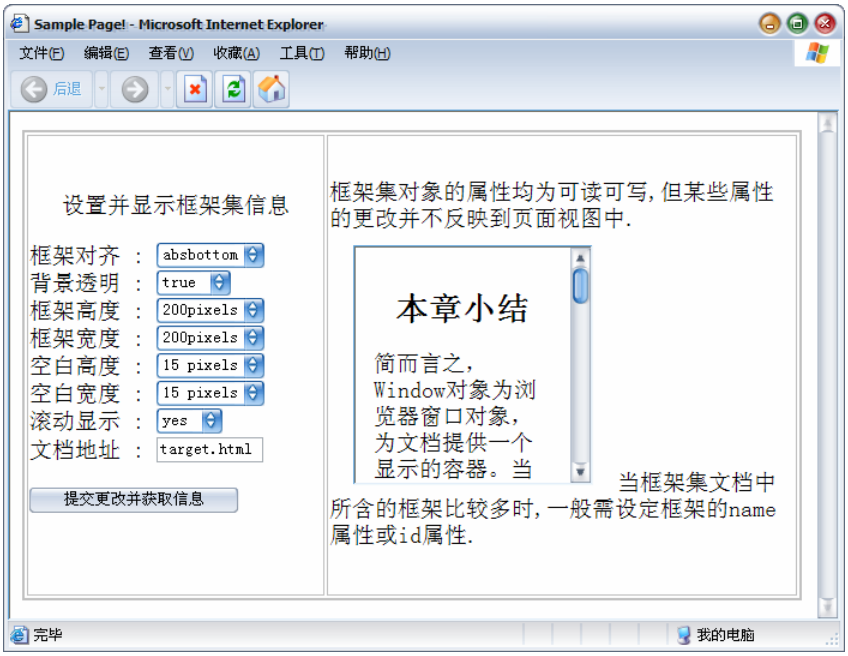

图 7.46 程序运行后出现的原始页面

在原始页面中修改各个下拉框的选项及文本框的内容后，单击“提交更改并获取信息”按钮，将弹出包含修改前后框架对应的 iframe 元素对象各属性值对比信息的警告框。如修改 “框架对齐”为“middle”、“框架高度”为 300、“框架宽度”为 350 等信息，并单击“提交更改并获取信息”按钮后，浏览器根据设置的属性值更新目标页面视图并弹出警告框如图 7.47 所示。

图 7.47 修改前后 iframe 元素对象各属性值对比

iframe 对象的 align 属性表示框架与其外内容的对齐方式，可以利用 JavaScript 脚本动态调整该属性来定位目标框架。表 7.11 列出了该属性的可能取值及简要说明：

表 7.11 iframe元素对象的align属性的可能属性值

取值	简要说明
absbottom	框架的底部与周围文本下方的虚线对齐
absmiddle	框架的中部与周围文本的top和absbottom值之间的中心点对齐
baseline	框架的滚动条按钮与周围文本的基线对齐
bottom	框架的滚动条按钮与周围文本的基线对齐（IE）
left	根据新设定的属性值刷新框架，同时使该框架与包含元素的左边缘对齐
middle	框架的垂直方向中心线与周围文本的虚垂直中心线对齐
right	根据新设定的属性值刷新框架，同时使该框架与包含元素的右边缘对齐
texttop	框架顶部与周围文本顶部最高点的虚线对齐
top	框架顶部与周围元素顶部最高点的虚线对齐

值得注意的是，除 contentDocument 和 contentWindow 属性为只读外，iframe 对象的其余属性均为可读可写，但并非所有的属性被改变后都能实时更新目标 iframe 框架，如控制滚动条显示与否的 scrolling 属性、控制框架与其外部的文档之间空白尺寸的 hspace 和 vspace 属性等，其更多的是作为一种只读属性而存在。

下面简要介绍顶级对象模型中的 Document 对象。

7.8 Document 对象

JavaScript 主要作为一门客户端脚本而存在，在与客户交互的过程中 Document 对象扮演着及其重要的角色。深入理解 Document 对象对编写跨浏览器的 Web 应用程序是 JavaScript 脚本程序员进阶的关键。

Document 对象在顶级对象模型中处于较低的层次，通俗地说，浏览器载入文档后，首先产生顶级对象模型中的 Window、Frame 等对象，且当该文档包含框架集时，一个 Window 对象下面可包含多个 Document 对象。

Document 对象及其组件相关的内容相当丰富，在“Document 对象”章节将对 Document 对象作专门的讲述。

7.9 本章小结

本章主要讲述了 Window 及相关顶级对象如 Frames、Navigator、Screen、History、Location、Document 等，并从实际应用的角度分析了其创建过程、属性、方法、操作实例等。并重点介绍了如何通过这些对象创建和管理浏览器窗口及文档中的框架。

Document 对象提供了与客户交互的平台，使用 JavaScript 脚本操作 Document 对象可以创建动态、交互性较强的页面，下一章将重点介绍 Document 对象的创建，以及如何使用 JavaScript 脚本调用其属性和方法来动态更新页面。