第 8 章 Document 对象

Document 对象在顶级对象模型中占据非常重要的地位，它可以更新正在装入和已经装入的文档，并使用 JavaScript 脚本访问其属性和方法来操作已加载文档中包含的 HTML 元素，如表单 form、单选框 radio、下拉框 checkbox 等，并将这些元素当作具有完整属性和方法的元素对象来引用。本章将重点讲述顶级对象模型中 Document 对象及与其相关的 body 元素对象的基础知识，如对象的创建、引用及与其它 HTML 元素对象之间的相互关系等。

8.1 对象模型参考

客户端浏览器载入目标 HTML 文档后，在创建顶级对象模型中其它顶级对象的同时，创建 Document 对象的实例，并将该实例指向当前的文档。当文档包含多个框架组成的框架集或者在该文档中由<iframe>和</iframe>标记对引入其它外部文档时，当前浏览器窗口就同时包含了多个 Document 对象。Web 程序开发人员根据对象之间的相对位置关系使用 JavaScript 脚本进行相关操作如对象定位、访问等。

Document 对象在文档结构模型中处于顶级层次，但较之如 Window 等其它顶级对象而言，该对象与客户端浏览器的关联程度比较小，而与所载入文档本身的关联程度较为紧密。图 8.1 从 Document 对象的角度出发，显示了它在文档对象模型的参考层次中所处的相对位置（NN4+和 IE4+文档结构模型通用）：

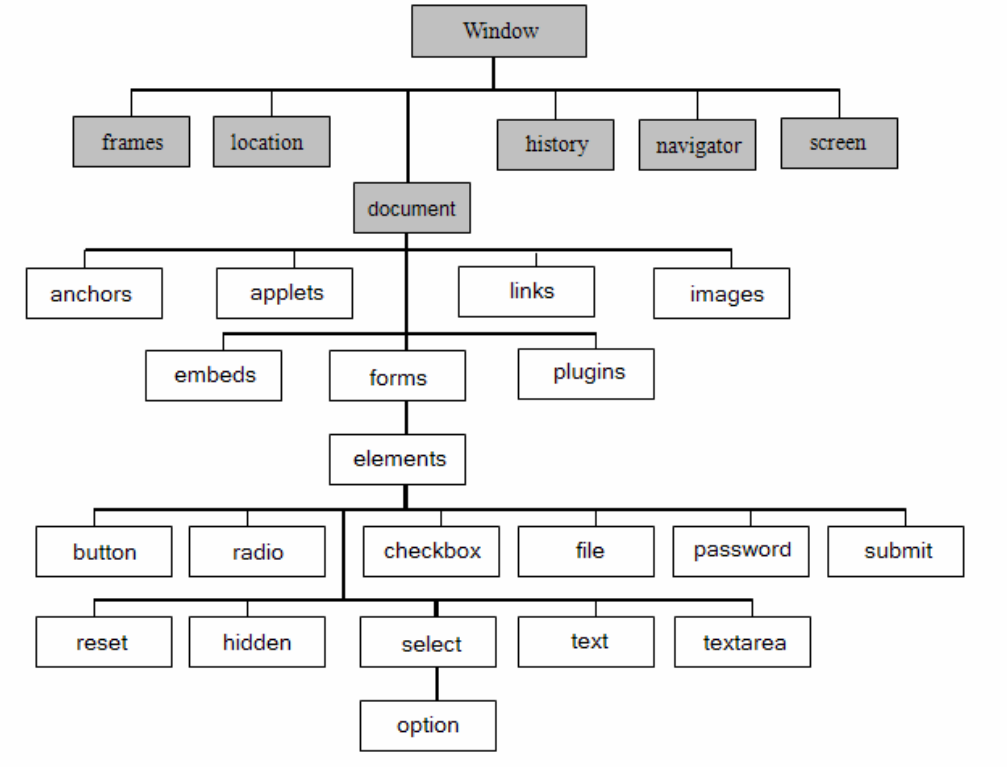

图 8.1 Document 对象模型参考

在上述的对象模型参考中，灰色表示的是 DOM 中的顶级对象，而 Document 对象所在层次之下的对象为目标文档包含的 HTML 元素对象。可见在文档中定位了 Document 对象之后，就可根据对象的层次关系操作其层次之下任意的元素对象。

> 注意：上面描述的对象模型中 frames 分别作为顶级对象和 Document 对象包含的元素对象而存在，因为当某文档包含框架集时，frames 对象作为该文档对应的 Document 对象的元素对象而存在。当框架集中某个框架载入另一个文档时，该文档对应的 Document 对象又作为 frames 对象下一层次的对象而存在。

8.2 Document 对象

Document 对象包括当前浏览器窗口或框架内区域中的所有内容，包含文本域、按钮、单选框、复选框、下拉框、图片、链接等 HTML 页面可访问元素，但不包含浏览器的菜单栏、工具栏和状态栏。

Document 对象提供多种方式获得 HTML 元素对象的引用，如在某目标文档中含有多个通过<form>和</form>标记对引入的表单，则可通过如下方式获得对该文档中 forms 对象数组长度信息的引用：

```
document.forms.length
document.getElementsByTagName("form").length
```

获取了对象数组信息后，就可以根据目标文档中该类型对象的相对位置定位某对象，如循环检索 forms 数组各表单的 name 属性的代码：

```
var MyForms=document.forms;
for(i=0;i<MyForms.length;i++)
{
  msg+="forms[" +i+ "].name : " +MyForms[i].name+ "\n";
}
```

代码运行后，将根据 forms 对象数组的长度信息遍历该数组并输出各表单 name 属性值。

8.2.1 获取目标文档信息

浏览器载入目标文档后，将根据文档标记的类型产生该类型的对象数组，并以标记元素载入的时间顺序进行数组下标分配。考察如下获取 Document 对象信息的代码，其中框架集文档“main.html”代码如下：

```
//源程序 8.1
<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" src="target.html" frameBorder="yes" borderColor="#c0c0c0">
</frameset>
</html>
```

该框架集文档包含右框架文档“target.html”和左框架文档“leftmain.html”，其中前者为普通测试文档，而后者为包含链接、图片、插件、表单等 HTML 页面可见元素的文档，其代码如下：

```
//源程序 8.2
<!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 msg="\nDocument 对象信息 : \n\n";
//获取 Document 对象信息
function GetInfo()
{
  //获取父文档的 frames 对象数组
  var MyFrames=parent.document.frames;
  msg+="父文档 frames 数组 : \n";
  msg+="长度 : "+MyFrames.length+"\n";
  for(i=0;i<MyFrames.length;i++)
  {
    msg+="frames[" +i+ "].name : " +MyFrames[i].name+ "\n";
  }
  //获取文档的 links 对象数组
  var MyLinks=document.links;
  msg+="links 数组 : \n";
  msg+="长度 : "+MyLinks.length+"\n";
  for(i=0;i<MyLinks.length;i++)
  {
    msg+="links[" +i+ "].name : " +MyLinks[i].name+ "\n";
  }
  //获取文档的 images 对象数组
  var MyImages=document.images;
  msg+="images 数组 : \n";
  msg+="长度 : "+MyImages.length+"\n";
  for(i=0;i<MyImages.length;i++)
  {
    msg+="images[" +i+ "].name : " +MyImages[i].name+ "\n";
  }
  //获取文档的 embeds 对象数组
  var MyEmbeds=document.embeds;
  msg+="embeds 数组 : \n";
  msg+="长度 : "+MyEmbeds.length+"\n";
  for(i=0;i<MyEmbeds.length;i++)
  {
    msg+="embeds[" +i+ "].name : " +MyEmbeds[i].name+ "\n";
  }
  //获取文档的 plugins 对象数组
  var MyPlugins=document.plugins;
  msg+="plugins 数组 : \n";
  msg+="长度 : "+MyPlugins.length+"\n";
  for(i=0;i<MyPlugins.length;i++)
  {
    msg+="plugins[" +i+ "].name : " +MyPlugins[i].name+ "\n";
  }
```

```
  //获取文档的 forms 对象数组
  var MyForms=document.forms;
  msg+="forms 数组 : \n";
  msg+="长度 : "+MyForms.length+"\n";
  for(i=0;i<MyForms.length;i++)
  {
    msg+="forms[" +i+ "].name : " +MyForms[i].name+ "\n";
  }
  //输出文档信息
  alert(msg);
}
//-->
</script>
</head>
<body>
<center>
<p>获取文档 Document 对象信息</p>
</center>
<table border=1 borderColor="#c0c0c0" align=center>
  <tr>
    <td align=middle>
      链接 :<br><br>
      <a href="target1.html" target="Display" name="linka">超级链接一</a><br>
      <a href="target2.html" target="Display" name="linkb">超级链接二</a><br>
      <a href="target3.html" target="Display" name="linkc">超级链接三</a><br>
    </td>
    <td align=middle>
      图片 :<br>
      <img src="image1.JPG" name="image1" width="84" height="98" border=1></img>
      <img src="image2.JPG" name="image2" width="84" height="98" border=1></img>
    </td>
  </tr>
  <tr>
    <td align=middle>
      音乐 :<br>
      <embed name="music" src="town.mid" units="pixels" height=130 width=165>
    </td>
    <td align=middle>
      多媒体 :<br>
      <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" codebase=
        "http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab
        #version=6,0,29,0" width="150" height="150">
        <param name="movie" value="ClockFlash.swf">
        <param name="quality" value="high">
        <embed src="ClockFlash.swf" quality="high" width="150"height="150"
          pluginspage="http://www.macromedia.com/go/getflashplayer"
          type="application/x-shockwave-flash" >
        </embed>
      </object>
    </td>
  </tr>
  <tr>
    <td align=left>
```

```
        <center>表单一 :</center><br>
        <form name="MyForm1">
          单选框 : <input type="radio" name="MyRadio" value="yes" checked>是
                   <input type="radio" name="MyRadio" value="no">否<br>
          下拉框 : <select name="MySelect" id="MySelect">
                     <option value="black" selected>黑色</option>
                     <option value="green">绿色</option>
                   </select><br>
        </form>
      </td>
      <td align=left>
        <center>表单二 :</center><br>
        <form name="MyForm2">
          复选框 : <input type="checkbox" name="MyCheckbox"value="yes" checked><br>
          文本框 : <input type="text" name="MyText" size="11" value="Welcome!"><br>
        </form>
      </td>
    </tr>
</table>
<center>
<form name="MyForm3">
    <input type="submit" name="MySure" value="获取信息" onclick="GetInfo()">
</form>
<center>
</body>
</html>
```

程序运行后，出现如图 8.2 所示的原始页面。

图 8.2 程序运行后的原始页面

鼠标单击上述原始页面中的“获取信息”按钮后，触发 GetInfo()函数收集当前文档对应的 Document 对象相关信息，并使用警告框输出，如图 8.3 所示。

图 8.3 当前文档的 Document 对象信息

由上图可以看出，浏览器载入文档后，根据 HTML 元素载入的顺序和类型生成对象数组并按顺序分配数组下标以便 JavaScript 脚本对各元素对象进行访问。对象数组生成后，可根据数组中各元素的相对位置或其标识符（name 属性等）来引用。如在上述实例中，表单 MyForm3 载入时顺序为 3，则可通过下面的方式引用该表单的 name 属性：

```
document.forms["MyForm3"].name
document.forms[2].name
```

当然，也可使用如下方式直接进行访问：

```
document.all.MyForm3.name
document.getElementByName("MyForm3")
```

其中，getElementByName()为 W3C DOM Level 1 中定义的标准方法，目前通用的浏览器版本都基本实现了 W3C DOM Level 1 规范。同样，如果设置了标记元素的 id 属性，也可使用 getElementById()方法来代替该方法。

综上所述，Document 对象一般的处理步骤如下：

- 获取文档的指定对象类型的目标数组，如 images、forms 数组等；
- 使用目标数组的长度为参数遍历数组，根据提供的属性值检索出目标在对象数组中的位置信息；
- 使用目标对象的位置信息访问该对象的属性和方法。

在 W3C 新版本中定义了更多访问 Document 对象中标记元素对象的通用方法，同时浏览器版本的更新也对其进行了大量的扩展，本章仅讨论通用浏览器版本上的 Document 对象的相关知识。

8.2.2 设置文档颜色值

Document 对象提供了几个属性如 fgColor、bgColor 等来设置 Web 页面的显示颜色，它们一般定义在<body>标记中，在文档布局确定之前完成设置。例如：

```
<body bgColor="white" fgColor="green" linkColor="red" alinkColor="blue" vlinkColor="purple">
```

其中 bgColor 属性可通过 JavaScript 脚本动态改变。Document 对象提供有关颜色的属性

分别代表如下：

- bgColor 表示文档的背景色；
- fgColor 表示文档中文本的颜色；
- linkColor 表示文档中未访问链接的颜色；
- alinkColor 表示文档中链接被单击时出现的颜色；
- vlinkColor 表示文档中已访问链接的颜色；

考察如下设置文档中各项颜色的实例代码：

```
//源程序 8.3
<!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">
<!--
//设置文档的颜色显示
function SetColor()
{
   document.bgColor="white";
   document.fgColor="green";
   document.linkColor="red";
   document.alinkColor="blue";
   document.vlinkColor="purple";
}
//改变文档的背景色为黑色
function ChangeColorOver()
{
   document.bgColor="black";
   return;
}
//改变文档的背景色为黑色
function ChangeColorOut()
{
   document.bgColor="white";
   return;
}
//-->
</script>
</head>
<body onload="SetColor()">
<center>
<br>
<p>设置颜色</p>
<a href="target.html">链接实例</a>
<form name="MyForm3">
    <input type="submit" name="MySure" value="改变背景色"
              onmouseover="ChangeColorOver()"
              onmouseOut="ChangeColorOut()">
</form>
<center>
</body>
```

```
</html>
```

该程序使用<body>标记内的 onload()事件调用 SetColor()方法来设置文档页面各项颜色的初始值如背景色 bgColor 为颜色字符串常量“white”。程序运行后，将出现如图 8.4 所示的页面。

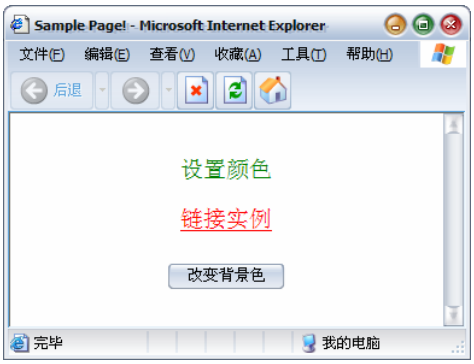

图 8.4 设置文档页面各项颜色的初始值

在上述页面中，鼠标移动到“改变背景色”按钮上时，触发 onmouseOver()事件调用 ChangeColorOver()函数来改变文档的背景颜色为黑色；当鼠标移离“改变背景色”按钮时，触发 onmouseOut()方法调用 ChangeColorOut()函数来改变文档的背景颜色为白色。

例如在页面中单击“链接实例”链接，并将鼠标移动到“改变背景色”按钮之上时，出现如图 8.5 所示的页面。

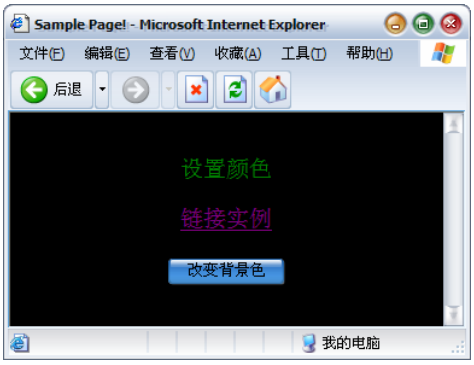

图 8.5 改变文档的背景色

在 HTML4 中，颜色有如下两种表示方式：

- 颜色字符串常量表示法：使用特定的字符串表示某种颜色，如字符串“blue”表示蓝色、“red”表示红色等。在 W3C 制定的 HTML 4.0 标准中，存在 16 个颜色字符串常量，详情请见表 8.1；
- RGB 原色表示法：RGB 是 Red、Green、Blue 三个词语的缩写，一个 RGB 颜色值由三个两位十六进制数字组成，分别代表各原色的强度。该强度为从 0 到 255 之间的整数值，如果换算成十六进制值表示，则范围从#00 到#FF。例如 RGB(255,255,255)表示白色，且用#FFFFFF 表示；RGB(0,0,0)表示黑色，且用#000000 表示。

在遵循 HTML4 规范的同时，各大浏览器厂商都扩展了在 HTML4 中预定义的颜色字符串常量，但 RGB 原色表示法可以在所有浏览器中得到正确的显示。在编写需跨浏览器环境工作的 HTML 文档时，要尽量使用 RGB 原色表示法以避免出现兼容性问题。

在 HTML4 中预定义的颜色字符串常量与 RGB 原色值之间存在一定对应关系，如表 8.1 所示。

表 8.1 颜色字符串常量与RGB原色值之间关系

字符串常量	RGB原色值	字符串常量	RGB原色值
aqua	#00ffff	navy	#000080
black	#000000	olive	#808000
blue	#0000ff	purple	#800080
fuchsia	#ff00ff	red	#ff0000
gray	#808080	silver	#c0c0c0
green	#008000	teal	#008080
lime	#00ff00	white	#ffffff
maroon	#800000	yellow	#ffff00

8.2.3 往文档写入新内容

Document 对象提供 open()用于打开新文档以准备执行写入操作，并提供 write()方法和 writeIn()方法用于写入新内容。这些动作完成后，可以使用 close()方法来关闭该文档。

open()方法的语法如下：

```
open([mimeType][,replace])
```

其中参数 mimeType 指定发送到窗口的 MIME 类型，如 text/html、image/jpeg 等。MIME 类型是在 Internet 上描述和传输多媒体数据的规范。在浏览器参数设置的辅助应用程序列表中已简单描述 MIME 类型，它是用斜杠分隔的一对数据类型。将某个 MIME 类型以参数传入 open()方法即是通知浏览器准备接收该类型的数据，接收完成后，浏览器调用其相关功能将该数据显示出来。

各浏览器支持的 MIME 类型会有所区别，一般来说，常见的 MIME 类型有超文本标记语言文本 text/html、普通文本 text/plain、RTF 文本 application/rtf、GIF 图形 image/gif、JPEG 图形 image/jpeg、au 声音文件 audio/basic、MIDI 音乐文件 audio/midi 或 audio/x-midi、RealAudio 音乐文件 audio/x-pn-realaudio、MPEG 文件 video/mpeg、AVI 文件 video/x-msvideo 等。open()方法默认（缺省）的 MIME 类型参数为 text/html，在文档载入浏览器前，使用脚本组合典型的数据类型，包括 HTML 页面上的任何图片。如果当前浏览器不支持以参数传入的特定 MIME 类型，则浏览器搜索支持该 MIME 类型的插入件。如果目标插入件存在，则浏览器装入该插入件，并在文档载入时将要写入的内容传入该插入件。open()方法接收的第二个可选参数 replace 表示待写入的目标文档是否替换浏览器历史列表中当前文档。

> 注意：在 IE3 中 open()方法不接收任何参数，IE4 中仅接收 MIME 类型为 text/html 的参数，IE5+和 NN4+支持第二个可选参数；若使用 open()方法打开文档成功，则该方法返回非 null 值，若由于某种原因打开文档失败，则返回 null 值。

使用 open()方法打开文档后，可调用 write()和 writeIn()方法往其中写入新内容并在浏览器窗口中显示出来。write()和 writeIn()方法本质上并无大的不同，唯一的区别在于后者在发送给文档的内容末尾添加换行符，下面两种表达方式等价：

```
document.write(targetStr+"<br>");
document.writeIn(targetStr);
```

实际上，一旦文档载入窗口或框架完成后，使用 write()和 writeIn()方法可在不重载或不重写整个页面的情况下改变文本节点和文本域 textarea 对象的内容。

一般情况下，脚本在当前文档收集用户输入（或动作）和浏览器环境信息，然后通过一定的算法产生表示另一个窗口（或框架）布局和内容的字符串变量，并使用 write()和 writeIn()

方法将目标字符串变量写入新窗口或者多框架窗口中的某个框架中。该种方式允许使用任意多个 document.write()和 document.writeIn()方法写入任意多个字符串变量，同时可通过一个组合字符串调用这两种方法来写入。实际中采用何种方法写入目标字符串取决于浏览器环境和脚本样式。

在组合字符串过程中，可通过加号“+”或逗号“,”将字符串连接起来，例如下列两种方式均合法：

```
document.write(targetStr+"<br>");
document.write(targetStr,"<br>");
```

考察如下将相关字符串写入框架集中指定框架的实例，演示了“学生注册程序”中如何收集学生相关信息并在目标框架显示。其中框架集文档“main.html”的代码如下：

```
//源程序 8.4
<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" src="target.html">
</frameset>
</html>
```

其中左框架载入的文档为“leftmain.html”，该文档收集学生信息如姓名、性别、年级和学号等。其代码如下：

```
//源程序 8.5
<!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">
<!--
function WriteContent()
{
   var msg="写入的新内容 ：<br><br>";
   //获取"姓名"字段信息
   var iName=document.MyForm.MyName;
   if(iName.value=="")
   {
      alert(""姓名"字段不能为空!");
      iName.focus();
   }
   else
   {
      msg+="姓名 ："+iName.value+"<br>";
   }
   //获取"性别"字段信息
   var iSex=document.MyForm.MySex;
   for(i=0;i<iSex.length;i++)
   {
      if(iSex[i].checked)
         msg+="性别 ："+iSex[i].value+"<br>";
```

```
  }
  //获取"班级"字段信息
  var iClass=document.MyForm.MyClass;
  for(i=0;i<iClass.length;i++)
  {
    if(iClass[i].selected)
      msg+="班级 : "+iClass[i].value+"<br>";
  }
  //获取"学号"字段信息
  var iNum=document.MyForm.MyNum;
  if(iNum.value=="")
  {
    alert(""学号"字段不能为空!");
    iNum.focus();
  }
  else
  {
    msg+="学号 : "+iNum.value+"<br>";
  }
  //写入新内容到右框架
  parent.frames[1].document.write(msg);
  //关闭文档
  parent.frames[1].document.close();
}
//-->
</script>
</head>
<body>
<center>
<br>
<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="male" checked>male
            <input type="radio" name="MySex" id="MySex" value="female">female</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>
              <option value="class 5" >Class 5</option>
              <option value="class 6" >Class 6</option>
            </select>
  <p>学号 : <input type="text" name="MyNum" id="MyNum" value="200104014104"></p>
  </p>
  <p>
  <input type="button" name="MySubmit" value="写新内容进右框架" onclick="WriteContent()">
  </p>
</form>
</center>
</body>
</html>
```

右框架为待写入新内容的目标框架，其载入的文档“target.html”代码为：

```
//源程序 8.6
<html>
<head>
   <title>Sample Page!</title>
</head>
<body>
   <p>待写入的目标框架</p>
</body>
</html>
```

浏览器载入“main.html”文档后，出现如图 8.6 所示的原始页面。

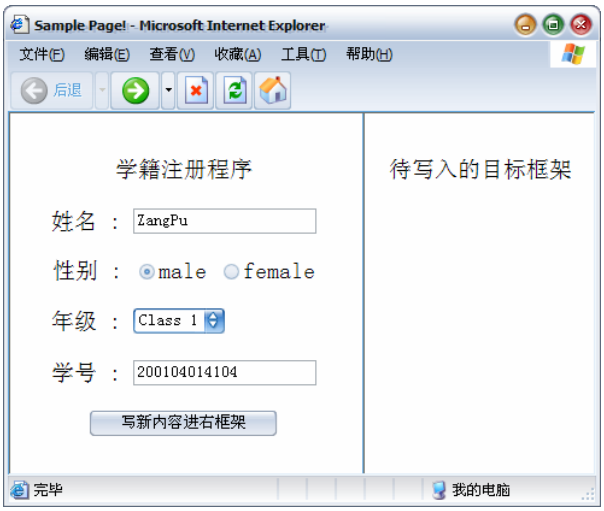

图 8.6 浏览器载入框架集文档后出现的原始页面

在上述原始页面中，单击“写新内容进右框架”按钮，触发左框架文档“leftmain.html”中 WriteContent()函数将通过当前框架文档 MyForm 表单收集的学生信息写入右框架文档中，同时更新页面，如图 8.7 所示。

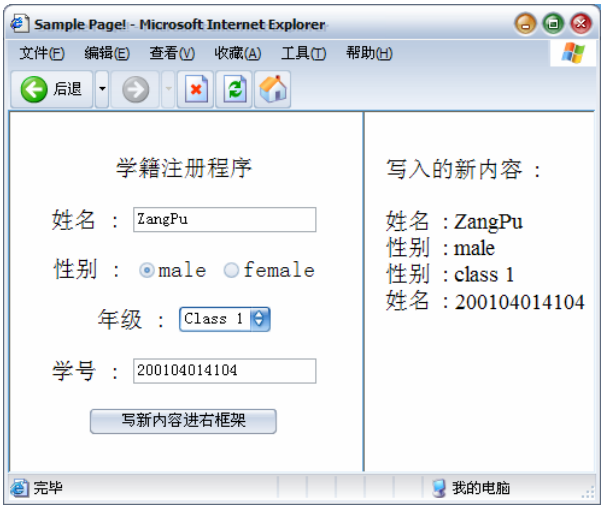

图 8.7 写入新信息后的框架集文档页面

查看此时右框架所载入文档的源代码，如下所示：

```
写入的新内容 ：<br><br>姓名 ：ZangPu<br>性别 ：male<br>性别 ：class 1<br>姓名 ： 20010401
4104<br>
```

除了可以往窗口（或框架）所载入文档中写入普通字符串外，还可写入 HTML 格式的代码来生成标准的 HTML 文档。修改上述实例中的“leftmain.html”文档中 WriteContent() 函数为下述形式：

```
//源程序 8.7
function WriteContent()
{
   var msg="<html><head><title>Sample Page!<\/title><\/head><body>";
   msg="写入的新内容 ：<br><br>";
   //获取"姓名"字段信息
   var iName=document.MyForm.MyName;
   if(iName.value=="")
   {
      alert(""姓名"字段不能为空!");
      iName.focus();
   }
   else
   {
      msg+="姓名 ："+iName.value+"<br>";
   }
   //获取"性别"字段信息
   var iSex=document.MyForm.MySex;
   for(i=0;i<iSex.length;i++)
   {
      if(iSex[i].checked)
         msg+="性别 ："+iSex[i].value+"<br>";
   }
   //获取"班级"字段信息
   var iClass=document.MyForm.MyClass;
   for(i=0;i<iClass.length;i++)
   {
      if(iClass[i].selected)
         msg+="班级 ："+iClass[i].value+"<br>";
   }
   //获取"学号"字段信息
   var iNum=document.MyForm.MyNum;
   if(iNum.value=="")
   {
      alert(""学号"字段不能为空!");
      iNum.focus();
   }
   else
   {
      msg+="学号 ："+iNum.value+"<br>";
   }
   msg+="<\/body><\/html>"
   //写入新内容到右框架
   parent.frames[1].document.write(msg);
   //关闭文档
   parent.frames[1].document.close();
}
```

单击左框架文档页面中“写新内容进右框架”按钮，将新内容写入右框架文档，其结果不变。但此时右框架所载入文档的源代码发生变化：

```
<html>
<head>
<title>Sample Page!</title>
</head>
<body>
写入的新内容 ：<br><br>姓名 ：ZangPu<br>性别 ：male<br>班级 ：class 1<br>学号 ：
200104014104<br>
</body>
</html>
```

可以看出，该文档为标准的 HTML 文档，但写入 HTML 标记如</head>时要注意写入的格式，如 document.write("<head></head>")语句将会出现错误，因为浏览器会将脚本标记的尾标记解释为进行写操作脚本的结束符。

可通过将尾标记分成几个部分的方式解决，可将上句改写如下：

```
document.write("<head><\/head>");
```

在使用组合字符串进行文档写操作时，容易出现难于觉察的错误，JavaScript 脚本程序初学者应尽量使用多个 write()和 writeIn()方法多次写入相关内容的方式，避免字符串相加时出现错误。

通过 Document 对象的 open()方法或者 write()方法打开一个到目标窗口或框架的输出流并往文档写入新内容的过程结束后，应使用其 close()方法来关闭该输出流。关闭输出流步骤相当重要，如果不关闭，则后续的写入操作会将新内容添加到该输出流对应的文档底部。

当 document.close()方法调用后，往指定窗口或框架添加的部分或全部数据才能正常显示，特别是 open()方法接收的 MIME 类型参数为 GIF 图形 image/gif 或 JPEG 图形 image/jpeg 时，使用 close()方法后图片才能正确显示出来。

8.2.4 常见属性和方法汇总

Document 对象提供一系列属性和方法操作目标文档，其属性分为包含文档附加信息的属性如标记文档背景色的 bgColor 等，以及文档结构模型中作为其属性的对象如表单元素对象 forms 等。表 8.2 列出了 IE 和 NN 浏览器上 Document 对象通用的属性、方法及浏览器版本支持情况。

表 8.2 Document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	alinkColor	表示<body>标记的alink属性	NN2+、IE3+
	anchor、anchors	表示文档中所有的锚点对象及形成的数组	NN2+、IE3+
	applet、applets	表示文档中所有嵌入的小程序及形成的数组	NN2+、IE3+
	area	表示文档中包含图形映射区的对象	NN2+、IE3+
	bgColor	表示<body>标记的bgColor属性值	NN2+、IE3+
	compatMode	表示在文档中DOCTYPE元素说明的兼容模式	NN7+、IE6+
	cookie	表示文档的cookie值	NN2+、IE3+
	domain	表示受当前文档信任的域名列表	NN3+、IE4+
	embeds	表示文档中所有插入件形成的数组	NN3+、IE4+
	fgColor	表示<body>标记的fgColor属性值	NN2+、IE3+
	form、forms	表示文档中所有表单对象及它们形成的数组	NN2+、IE3+
	image、images	表示文档中所有图片对象及它们形成的数组	NN3+、IE4+
	lastModified	表示文档最后的修改时间	NN2+、IE3+
	link、links	表示文档中所有链接对象及它们形成的数组	NN2+、IE3+

	linkColor	表示<body>标记的linkColor属性值	NN2+、IE3+
	plugin、plugins	表示文档中所有插入件对象及它们形成的数组	NN4+、IE4+
	referrer	为文档提供一个链接文档的URL	NN2+、IE3+
	title	表示文档的标题栏文本内容	NN2+、IE3+
	URL	表示文档的URL地址	NN3+、IE4+
	vlinkColor	表示<body>标记的vlinkColor属性值	NN2+、IE3+
方法	clear()	清除当前文档的内容	NN2+、IE3+
	close()	关闭用于创建当前文档对象的流	NN2+、IE3+
	createAttribute(attributeStr)	创建新的attribute对象并引用该对象	NN6+、IE6+
	createComment(commentStr)	创建新注释节点对象并引用该对象	NN6+、IE6+
	createElement(elementStr)	创建以参数elementStr为名称的HTML元素	NN6+、IE6+
	createTextNode(nodeStr)	创建以参数nodeStr为名称的文本节点对象	NN6+、IE5+
	exeCommand(commandStr)	执行以参数commandStr标识的命令	NN7+、IE4+
	getElementById(idStr)	根据元素的id属性引用文档中任意元素	NN6+、IE5+
	getElementByName(nameStr)	根据元素的name属性引用文档中任意元素	NN6+、IE5+
	open([mimeType][,replace])	用可选MIME类型的参数mimeType打开用于创建当前文档对象的流，repalce参数用于取代历史清单中的当前文档	NN2+、IE3+
	queryCommandEnabled(str)	显示适合调用的对象是Document还是TextRange	NN7+、IE4+
	queryCommandIndtem(str)	显示命令是否处于不确定状态	NN7+、IE4+
	queryCommandCommandState (str)	显示命令是处于完成状态（true）、正在执行状态（false）、还是不确定状态（null）	NN7+、IE4+
	queryCommandSupported(str)	显示当前浏览器是否支持指定的命令	NN7+、IE4+
	queryCommandText(str)	返回命令执行完毕以结果返回的任何文本	NN7+、IE4+
	queryCommandValue(str)	返回命令执行完毕返回的结果（如果存在）	NN7+、IE4+
	write(expr1[,expr2,…,exprn])	将表达式expr1,expr2,…,exprn写入当前文档	NN2+、IE3+
	writeIn(expr1[,expr2,…exprn])	将表达式expr1,expr2,…,exprn写入当前文档并在结尾加上换行符	NN2+、IE3+

上表中关于元素节点对象的相关内容如 createTextNode()、createElement()等方法已在“文档对象模型(DOM)”章节详细叙述，此处不再累述。

Document 对象提供的属性和方法主要用于设置浏览器当前载入文档的相关信息、管理页面中已存在的标记元素对象、往目标文档添加新文本内容、产生并操作新的元素对象等方面。在包含框架集的文档中，要注意 Document 对象引用的层次关系，并通过该层次关系准确定位目标对象并调用 Document 对象的属性和方法进行相关操作。

关于 Document 对象下一层次中的对象如 forms 对象、links 对象、images 对象等将在后续的章节进行专门的叙述。下面讲述与 Document 对象紧密相连的 body 元素对象。

8.3 body 元素对象

Document 对象在 HTML 文档中并没有使用任何显式标记来描述，但 JavaScript 脚本将其作为文档设置的入口，这些设置的内容又作为 body 元素的内在属性在 HTML 中描述。根据 DOM 中的节点模型概念，可将 body 元素对象作为<body>标记的引用。首先考虑如下简单的 HTML 代码：

```
<body bgColor="red" fgColor ="blue">
<p>Contents</p>
<table>
  ...
</table>
<form name="MyForm">
  ...
</form>
```

```
</body>
```

如上述的文档背景颜色 bgColor、文本颜色 fgColor 等均可以作为 body 元素对象的属性来调用；同时，body 标记元素对象也可以作为其他标记元素对象如<table>、<form>等的 HTML 容器而存在。

body 元素对象具有 Document 对象的大部分功能，但不具有 title、URL 等属性，主要包含与文档页面相关的属性和方法，如设置文档背景图片的 background 属性、控制页面滚动条的 scroll 属性等。

8.3.1 获取 body 元素对象信息

如前所述，Document 对象提供了表示文档背景、文本、链接等颜色的属性，body 对象也提供了这几个属性的别名来访问文档中的各项与颜色相关的内容，如 body 元素对象的 aLink 属性、link 属性和 vLink 属性分别对应于 Document 对象的 alinkColor 属性、linkColor 属性和 vlinkColor 属性，而 text 属性则对应于 Document 对象的 fgColor 属性。

考察如下获取目标文档 body 元素对象信息的实例代码：

```
//源程序 8.8
<!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">
<!--
//获取 body 元素对象信息
function GetInfo()
{
   var msg="\n 获取 body 元素对象信息 : \n\n";
   msg+="document.body.aLink = " +document.body.aLink+ "\n";
   msg+="document.body.bgColor = " +document.body.bgColor+ "\n";
   msg+="document.body.link = " +document.body.link+ "\n";
   msg+="document.body.text = " +document.body.text+ "\n";
   msg+="document.body.vLink = " +document.body.vLink+ "\n";
   msg+="document.body.background = " +document.body.background+ "\n";
   alert(msg);
}
//-->
</script>
</head>
<body aLink="blue" bgColor="white" link="red" text="black" vLink="purple"
        background="beijing.jpg">
<center>
<br>
<a href="#">文本链接测试</a>
<form name="MyForm">
   <p>
   <input type="button" name="MyGet" value="获取 body 元素对象信息" onclick="GetInfo()">
   </p>
</form>
</center>
```

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

上述代码在<body>标记内初始化了文档背景、文本、链接等颜色及文档的背景图片。程序运行后，出现如图 8.8 所示的原始页面。

图 8.8 显示 body 元素对象信息的原始页面

在原始页面中单击“获取 body 元素对象信息”按钮，触发“GetInfo()”函数，收集当前文档的 body 元素对象的相关信息后，弹出如图 8.9 所示的警告框。

图 8.9 显示当前文档的 body 元素对象相关信息

body 元素对象中与颜色相关的属性如 bgColor、text 等与 body 元素的 HTML 属性等同，而不是与旧属性名相联系。

通过 body 元素对象的 background 属性设置文档页面背景图片后，浏览器使用目标图片覆盖背景颜色，可以通过如下脚本动态去除该背景图片并显示背景色：

```
document.body.background= "";
```

以上列举的属性在 NN6+和 IE4+浏览器上均获得完善的支持。

8.3.2 常见属性和方法汇总

body 元素对象提供的属性和方法为数不多，但提供了快捷设置文档页面的途径。表 8.3 列出了 body 元素对象常见的属性、方法和浏览器版本支持情况。

表 8.3 body元素对象常见属性和方法汇总

类型	项目	简要说明	浏览器版本
属性	aLink	表示文档中文本链接被单击后的颜色	NN6+、IE4+
	background	表示文档的背景图片	NN6+、IE4+

	bgColor	表示文档的背景色	NN6+、IE4+
	bgProperties	标识文档滚动时页面背景图片是固定还是随文档移动，可选值scroll（表示滚动）和fixed（表示固定）	IE4+
	bottomMargin	保存文档内容与浏览器窗口或框架底部的距离	IE4+
	leftMargin	保存文档内容与浏览器窗口或框架左边的距离	IE4+
	link	表示文档中未访问文本链接的颜色	NN6+、IE4+
	rightMargin	保存文档内容与浏览器窗口或框架右边的距离	IE4+
	text	表示文档中文本的颜色	NN6+、IE4+
	topMargin	保存文档内容与浏览器窗口或框架顶部的距离	IE4+
	vLink	表示文档中已访问文本链接的颜色	NN6+、IE4+
	noWrap	标识是否将文档中文本限制在窗口或框架的宽度内，参数为布尔值“true”或“false”	IE4+
	scroll	标识是否隐藏文档的滚动条，参数为布尔型字符串“yes”或“no”	IE4+
	scrollLeft	返回页面左边与水平滚动条左端之间的距离	NN7+、IE4+
	scrollTop	返回页面顶部与垂直滚动条顶部之间的距离	NN7+、IE4+
方法	createControlRange()	返回处于编辑模式下当前文档选定范围内所有控件形成的数组，常规视图下返回空数组	IE5+
	createTextRange()	返回包含body元素的HTML文本和body文本对应的初始TextRange对象	IE4+
	doScroll(ScrollAction)	模拟滚动条上的用户动作，以ScrollAction标识用户的动作，如PageDown、Left、PageUp等	IE5+

在提供上述属性和方法基础上，body 元素对象提供事件处理程序 onscroll 响应用户的动作，如鼠标移动滚动条到底部时调用 Window 对象的 scroll()方法将当前浏览器窗口移动到指定位置等。

8.4 本章小结

本章主要介绍了顶级对象模型中与文档紧密相关的 Document 对象和 body 元素对象，重点讲述了对象模型中 Document 对象和 body 元素对象的层次关系及基础知识，如对象的创建、引用及与其它 HTML 元素对象之间的相互关系等。理解了 HTML 文档中元素对象如 form、image 等都作为 Document 对象层次之下的元素对象而存在，并各自具有独特的属性和方法用于文档界面的设置。

从下章开始，将进入 DOM 层次规范中 Document 对象层次以下的元素对象如 anchor、link 等对象的基本概念的学习，并通过实例让读者熟悉其属性和方法，并深刻体会浏览器版本与它们之间的依存关系。