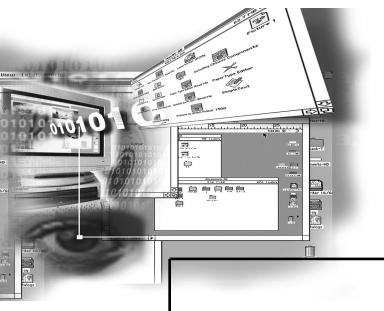


第 4 章　文档对象

Document（文档）对象是一种最基本的浏览器对象，它代表当前在浏览器窗口中打开的文档。使用 document 对象可以访问页面上的各种元素，通过控制这些页面元素可以实现需要的效果或功能。

本章介绍有关浏览器中 document 对象的基础知识和实际应用，主要内容包括：

- 浏览器对象简介
- document 对象的属性
- document 对象的事件
- document 对象的方法

4.1　浏览器对象简介

在本书第 2 章中已经说明，JavaScript 中包括两种对象——内置对象和浏览器对象。内置对象包括一些常用的通用对象，例如数组对象 Array、字符串对象 String、日期对象 Date 等，这些都已经在第 2 章中介绍过；而浏览器对象是指支持 JavaScript 的浏览器在装入 Web 页面时创建出的多个 JavaScript 对象，可以通过这些对象访问 Web 页面中的各种元素，获得相应的操作效果，例如我们经常用到的 document 对象、form 对象等。

从本章开始我们将陆续介绍各种常用的浏览器对象，利用这些对象可以实现各种客户端应用。

4.1.1　文档对象模型

文档对象模型（Document Object Model，简称 DOM）是用于表示 HTML 元素以及 Web

浏览器信息的一个模型，它使脚本能够访问 Web 页上的信息，并可以访问诸如网页位置等特殊信息。通过操纵文档对象模型中对象的属性并调用其方法，可以使脚本按照一定的方式显示 Web 页并与用户的动作进行交互。

对于不同的脚本语言，通常都具有一个 DOM 的子集，以便在特定的脚本语言中实现对象模型。例如，JavaScript 语言中就有一个对象模型。对于 Internet Explorer，Microsoft 公司专门为其创建了一个对象模型。使用为浏览器创建对象模型的方式使得对象模型与语言无关，从而可以获得更强的可扩展性。

JavaScript 对象模型和 IE 对象模型非常相象，它们包含相似的对象和事件，反映了如图 4.1 所示的对象层次结构。

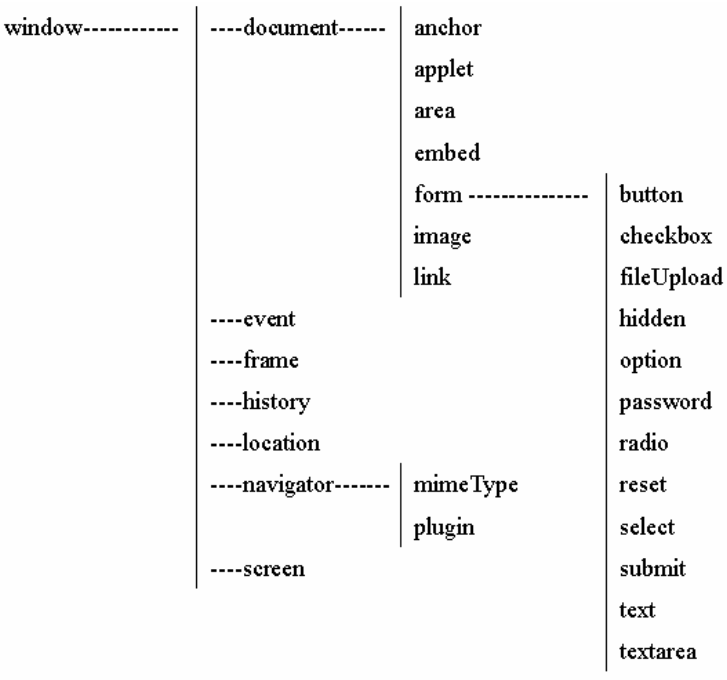
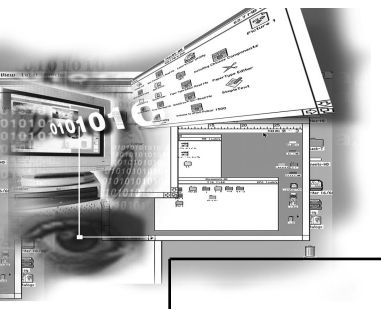


图 4.1　文档对象模型

在该层次结构中，最高层的对象是窗口对象（ window），它代表当前的浏览器窗口；之下是文档（ document）、事件（ event）、框架（ frame）、历史（ history）、地址（ location）、浏览器（ navigator）和屏幕（ screen）对象；在文档对象之下包括表单（ form）、图像（ image）和链接（ link）等多种对象；在浏览器对象之下包括 MIME 类型对象（ mimeType）和插件（ plugin）对象；在表单对象之下还包括按钮（ button）、复选框（ checkbox）、文件选择框（ fileUpload）等多种对象。



4.1.2 对象引用方法

了解了浏览器对象的层次结构之后，我们就可以用特定的方法引用这些对象，以便在脚本中正确地使用它们。

在JavaScript中引用对象的方式与典型的面向对象方法相同，都是根据对象的包含关系，使用成员引用操作符（.）一层一层地引用对象。例如，如果要引用 document 对象，应使用 window.document；如果要引用 location 对象，应使用 window.location。由于 window 对象是默认的最上层对象，因此引用它的子对象时，可以不使用 window.，也就是说，可以直接用 document 引用 document 对象，用 location 引用 location 对象。

当引用较低层次的对象时，一般有两种方式：使用对象索引或使用对象名称（或 ID）。例如，如果要引用文档中的第一个表单对象，则可以用 document.forms[0]来引用；如果该表单的 name 属性为 form1（或者 ID 属性为 form1），则可以用 document.forms["form1"]或直接用 document.form1 来引用该表单。同样，如果在名称为 form1 的表单中包括一个名称为 myText 的文本框，则可以用 document.form1.myText 来引用该文本框对象。

说明：由于 name 和 ID 属性的值使用相同的名字空间，因此最好只使用其中一种，以免造成混淆。也就是说，在 FORM 标记符中，最好只指定 name 属性和 ID 属性中的一种，并且不论是名称还是 ID，都不能重复。

对应于不同的对象，通常还有一些特殊的引用方法。例如，如果要引用表单对象中包含的对象，可以使用 elements 数组；如果要引用文档对象中包含的某个标记符对象（例如 P 对象），可以使用 document 对象的 all 属性，等等。对于这些特殊的引用方法，我们将在遇到时具体说明。

4.2 document 对象的属性

document 对象代表当前浏览器窗口中的文档，使用它可以访问到文档中的所有其他对象（例如图像、表单等），因此该对象是实现各种文档功能的最基本对象。本节首先介绍 document 对象的各种属性，之后的两节将分别介绍 document 对象的事件和方法。

4.2.1　属性列表

表 4.1 列出了 document 对象的各种属性以及相应的说明。

表 4.1　document 对象的属性

属　性	说　明
alinkColor	表示活动超链接的颜色
all	表示文档中所有 HTML 标记符的数组
anchors	表示文档中所有锚的数组，锚是指带有 name 属性的 A 对象
applets	表示文档中所有 JAVA 小应用程序
bgcolor	表示文档的背景颜色
cookie	表示与文档相关的 Cookie
domain	表示提供文档的服务器域
embeds	表示文档中所有嵌入对象的数组
fgColor	表示文档的前景颜色
forms	表示文档中所有表单的数组
images	表示文档中所有图像的数组
lastModified	表示文档的最后修改日期
linkColor	表示未被访问的超链接的颜色
links	表示文档中所有超链接的数组，超链接是指带有 href 属性的 A 或 Area 对象
referrer	表示链接到当前文档的文档的 URL，也就是说，如果当前文档是从另外一个文档调用而进入的，则当前文档的 referrer 属性为该文档的 URL
title	表示文档的标题
URL	表示文档的 URL
vlinkColor	表示已被访问的超链接的颜色

说明：虽然本书主要针对 Internet Explorer，但为了使所介绍的内容具有更大范围的兼容性，因此不论是属性还是方法，都尽量不介绍仅适用于 IE 的内容——除非必须如此（例如 all 属性）。

4.2.2　用 all 属性访问 HTML 元素

在 document 的属性中，有一个属性非常特殊和重要，它就是 all 属性。通过该属性，用户可以访问文档中的所有 HTML 元素对象。

使用 all 属性访问 HTML 元素对象有三种方法：

- 通过索引或名称直接引用；
- 使用 item() 方法；
- 使用 tags() 方法。

1．直接引用

用户可以直接用索引或名称的方式引用文档中的任意一个元素对象。例如，如果要引

用文档中的第 5 个元素对象，则可以使用 document.all[4]；如果该对象的 name 属性为 myTag5，则也可以使用 document.all["myTag5"]进行引用。

以下示例显示了如何使用这种引用方式，代码如下：

```
<HTML>
<HEAD>
  <TITLE>使用 all 引用对象</TITLE>
</HEAD>
<BODY>
<FORM name=form1>
请输入一个数字：<P>
  <INPUT name="myText" >
  <INPUT type=button value="数据处理">
</FORM>
<HR>
<H2>以下是本文档中出现的所有 HTML 标记：</H2>
<SCRIPT LANGUAGE = "JavaScript" TYPE="text/javascript">
<!--
for(i=0; i<document.all.length; i++)  /*由于 all 属性本身是一个数组，因此具有 length 属性*/
 document.write(document.all[i].tagName+",")  //tagName 属性表示元素的标记符名称
//-->
</SCRIPT>
</BODY>
</HTML>
```

执行该段代码的效果如图 4.2 所示。

说明：不论在文档中是否使用了以下标记符（这些标记符在 HTML 文档中可以省略），由 all 数组返回的元素中始终包含它们：HTML、HEAD、TITLE 以及 BODY。另外，与开始标记符相匹配的结束标记符不再作为单独的对象。

2．使用 item()方法

all 对象的 item()方法提供了用 HTML 元素对象的名称或 ID 访问对象的途径。该方法的返回值为具有指定名称或 ID 的元素，如果找到多个具有相同名称或 ID 的元素，则返回一个数组。

图 4.2　使用 all 属性引用对象

该方法的语法为：document.all.item("string")

以下示例使用 item()方法查找文档中所有名称或 ID 为"test"的元素对象，代码如下：

```
<HTML>
<HEAD>
  <TITLE>使用 item() 方法</TITLE>
</HEAD>
<BODY>
<FORM name=form1>
请输入一个数字：<P>
  <INPUT name="test" >
  <INPUT type=button value="数据处理">
</FORM>
<HR>
<H2 ID="test">以下是本文档中名称或 ID 为 "test" 的所有 HTML 标记：</H2>
<SCRIPT LANGUAGE = "JavaScript" TYPE="text/javascript">
<!--
var result=document.all.item("test");
if(result!=null)
{
if(result.length!=null)
 for(i=0; i<result.length; i++)
   document.write(result[i].tagName+",")
}
else
  document.write("没有发现名称或 ID 为 'test' 的元素！")
//-->
```

```
</SCRIPT>
</BODY>
</HTML>
```

执行该段代码的效果如图 4.3 **所示。**

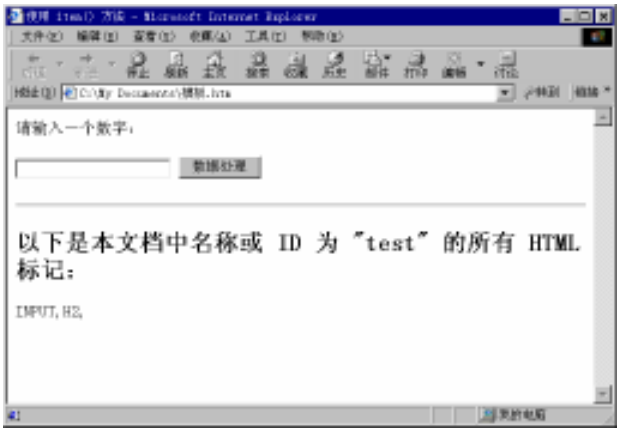


图 4.3　**使用** item()**方法**

3．使用 tags()方法

all **对象的** tags()**方法用来返回文档中具有特定标记符类型的所有** HTML **元素对象，返回值为一个数组。**

该方法的语法为：document.all.tags("tagName")

以下示例使用 tags() **方法查找文档中包含的段落数，代码如下：**

```
<HTML>
<HEAD>
  <TITLE>使用 tags() 方法</TITLE>
</HEAD>
<BODY>
<P>tags() 方法用来返回文档中具有特定标记符类型的所有 HTML 元素对象。</P>
<P>返回值为一个数组。</P>
<P>以下示例使用 tags() 方法查找文档中包含的段落数，代码如下。</P>
<HR><BR>
<SCRIPT LANGUAGE = "JavaScript" TYPE="text/javascript">
<!--
  document.write("<H2>本文档中包含"+document.all.tags("P").length+"个段落。</H2>")
```

```
//-->
</SCRIPT>
</BODY>
</HTML>
```

执行该段代码的效果如图 4.4 所示。

图 4.4　使用 tags()方法

4.2.3　其他属性示例

本小节中的示例显示了除 all 以外其他几个常用属性的用法。

1．示例 1

本示例列举了文档的标题信息以及其中包含的对象信息，代码如下：

```
<HTML>
<HEAD><TITLE>显示文档信息</TITLE></HEAD>
<BODY>
<A NAME="#top"></A>
<A HREF="http://nonexist.yeah.net"><IMG SRC="./images/pic1.jpg"></A>
<IMG SRC="./images/jieba.gif">
<FORM>
 <INPUT TYPE="BUTTON" VALUE="示例按钮">
</FORM>
<A HREF="http://zhaofengnian.yeah.net">作者主页</A>
<A HREF="#top">返回页首</A>
<HR>
```

```
<SCRIPT LANGUAGE = "JavaScript" TYPE="text/javascript">
<!--
document.write("<H3>本文档的统计信息如下：</H3>")
document.write("本文档的标题为："+document.title+"<BR>")
document.write("本文档的最后修改时间为："+document.lastModified+"<BR>")
document.write("本文档中包含 <B>"+document.links.length+" </B>个超链接
<BR>")
document.write("本文档中包含 <B>"+document.anchors.length+" </B>个锚点
<BR>")
document.write("本文档中包含 <B>"+document.forms.length+" </B>个表单
<BR>")
document.write("本文档中包含 <B>"+document.images.length+" </B>个图像
<BR>")
document.write("本文档中包含 <B>"+document.applets.length+" </B>个 Java
小应用程序<BR>")
document.write("本文档中包含 <B>"+document.embeds.length+" </B>个嵌入对象
<P>")
// -->
</SCRIPT>
</BODY>
</HTML>
```

该段代码的显示效果如图 4.5 所示。

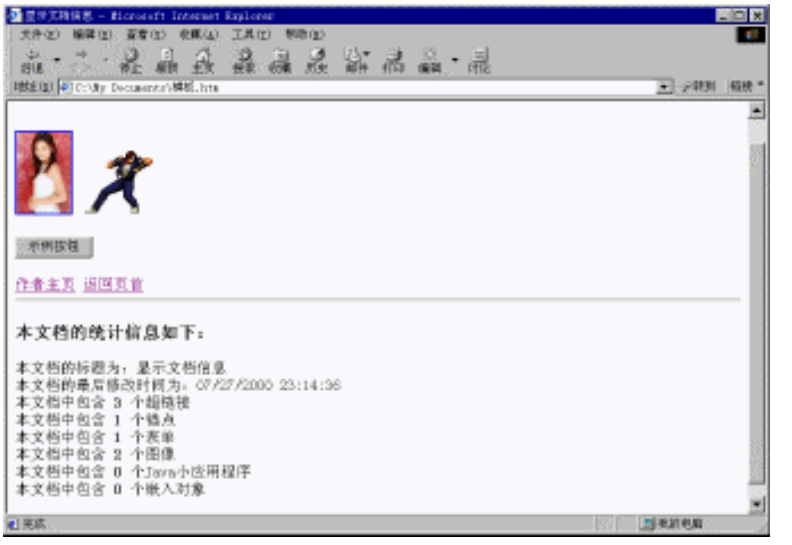


图 4.5　显示文档信息

2．示例 2

本示例显示了如何使用几个表示颜色的属性，代码如下：

```
<HTML>
<HEAD>
<TITLE>使用与颜色相关的属性</TITLE>
<SCRIPT LANGUAGE = "JavaScript" TYPE="text/javascript">
<!--
function changeColor()
{
document.bgColor=document.myForm.myBGColor.value;
document.fgColor=document.myForm.myFGColor.value;
document.linkColor=document.myForm.myLinkColor.value;
document.vlinkColor=document.myForm.myVLinkColor.value;
document.alinkColor=document.myForm.myALinkColor.value;
}
// -->
</SCRIPT>
</HEAD>
<BODY>
<H2>请在以下文本框中输入各种与页面有关的十六进制颜色值（例如：#00ffff）：</H2>
<FORM name="myForm">
<TABLE>
<TR>
  <TD>背景颜色：<TD><INPUT name="myBGColor" value="#ffffff"><P>
<TR>
  <TD>前景颜色：<TD><INPUT name="myFGColor" value="#000000"><P>
<TR>
  <TD>未访问过的超链接的颜色：<TD><INPUT name="myLinkColor"><P>
<TR>
  <TD>已访问过的超链接的颜色：<TD><INPUT name="myVLinkColor"><P>
<TR>
  <TD>活动超链接的颜色：<TD><INPUT name="myALinkColor"><P>
</TABLE>
  <INPUT TYPE="BUTTON" VALUE="单击此按钮更改文档的颜色设置！"
   onclick="changeColor()"><P>
</FORM>
<HR>
<A href="nonexist.com.cn">超链接示例</A>
</BODY>
```

```
</HTML>
```

该段代码的显示效果为：在各文本框中设置了特定的颜色后，单击指定按钮可以应用所设置的颜色；如果不指定某选项的颜色，则该颜色设置保持原来的默认值；如图 4.6 所示。

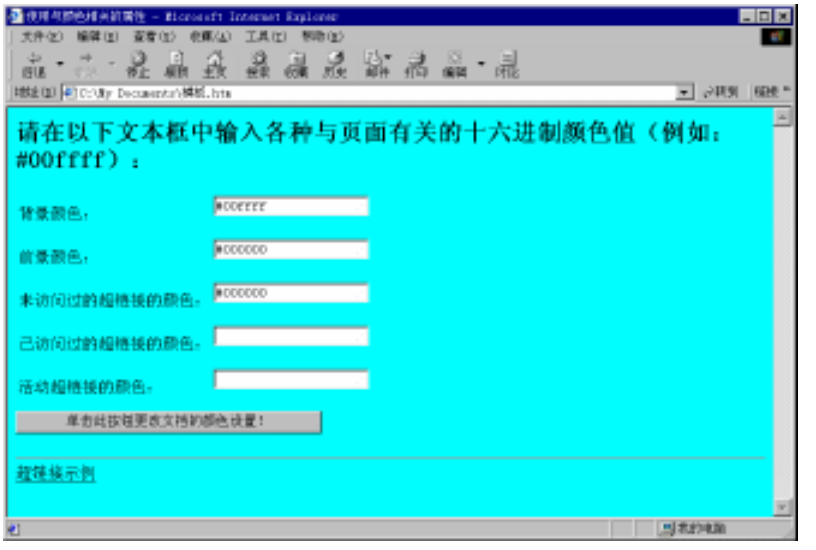


图 4.6　使用与颜色相关的属性

4.3　document 对象的事件

document 对象对应于整个网页对象，它接收发生在网页上的各种事件，同时也是事件起泡机制中的最顶层对象。在上一章中介绍 event 事件时，我们已经介绍过如何响应一些基本的事件，本节对 document 对象的事件响应问题作进一步说明。

4.3.1　处理键盘事件

document 对象可以处理 3 个基本的键盘事件 keydown、keypress 和 keyup，其中 keydown 和 keypress 都是当用户在文档中按下任意键时发生，而 keyup 是在用户释放某键时发生。

在实际应用过程中，最常用的是 keydown 事件，并且经常与 event 事件一起使用。以下示例显示了如何响应用户按下的组合键事件，代码如下：

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

```
<TITLE>处理键盘事件</TITLE>
<SCRIPT Language ="JavaScript" TYPE="text/javascript">
<!--
function keyHandler()
{
var realKey=String.fromCharCode(event.keyCode);
/* String 对象的 fromCharCode 方法用于返回参数所代表的字符，即将代码转换为字符。
*/
if(event.ctrlKey && event.altKey && event.shiftKey)
  document.myForm.result.value="Ctr+Alt+Shift+"+realKey;
else if (event.ctrlKey && event.altKey)
  document.myForm.result.value="Ctr+Alt+"+realKey;
else if(event.ctrlKey && event.shiftKey)
  document.myForm.result.value="Ctrl+Shift+"+realKey;
else if(event.altKey && event.shiftKey)
  document.myForm.result.value="Alt+Shift+"+realKey;
else if(event.ctrlKey)
  document.myForm.result.value="Ctrl+"+realKey;
else if(event.shiftKey)
  document.myForm.result.value="Shift+"+realKey;
else if(event.altKey)
  document.myForm.result.value="Alt+"+realKey;
}
//-->
</SCRIPT>
</HEAD>
<BODY onkeydown="keyHandler()">
<DIV align="center">
 <H2>请按任意组合键（可以用 Ctrl、Shift 和 Alt 键）</H2>
<BR><BR>
 <FORM name=myForm>
您刚才按下了：
  <INPUT name=result value="">
 </FORM>
</DIV>
</BODY>
</HTML>
```

当执行以上代码时，用户可以先按下 Ctrl、Shift 和 Alt 键的任意组合，然后按任意其他键，则所按的组合键将在文本框中显示，如图 4.7 所示。如果所按下的组合键碰巧又是浏览

器定义的快捷键，则在文本框中显示该组合键的同时还将执行浏览器的操作。例如，如果按 Alt+F 键，则会激活浏览器的“ 文件 ”菜单。

说明：本示例显示了如何处理键盘组合键，读者可以根据自己的实际需要添加自己的处理逻辑。

图 4.7　处理键盘事件

4.3.2　处理鼠标事件

document 对象通常处理以下四个基本的鼠标事件：click、dblclick、mousedown 和 mouseup。

以下示例显示了处理鼠标事件的基本机制，代码如下：

```
<HTML>
<HEAD>
<TITLE>处理鼠标事件</TITLE>
<SCRIPT Language ="JavaScript" TYPE="text/javascript">
<!--
function mouseDownHandler()
{
document.myForm.start.value=event.x+","+event.y;
}
function mouseUpHandler()
{
```

```
document.myForm.end.value=event.x+","+event.y;
}
//-->
</SCRIPT>
</HEAD>
<BODY onmousedown="mouseDownHandler()" onmouseup="mouseUpHandler()">
<DIV align="center">
 <H2>以下文本框中将显示用户拖放鼠标的起始点和终点：</H2>
<BR>
 <FORM name=myForm>
开始点：<INPUT name=start value=""><P>
终点：<INPUT name=end value="">
 </FORM>
</DIV>
</BODY>
</HTML>
```

这段代码的执行效果是：用户按住鼠标按钮（可以是左键或右键）不放，然后拖动到另外一个位置后释放按钮，则可以在文本框中显示鼠标拖放的起始点和终点，如图 4.8 所示；如果用户在某位置单击鼠标按钮，则起始点和终点的坐标都相同。

图 4.8　处理鼠标事件

4.3.3 处理加载卸载事件

对于 document 对象还包括两个常用的事件：onload 和 onunload，分别在文档装载完毕和卸载（或从当前页跳转到其他页）完毕时发生。

以下示例显示了这两个事件的作用，代码如下：

```
<HTML>
<HEAD>
  <TITLE>处理加载卸载事件</TITLE>
</HEAD>
<BODY onload="alert('欢迎光临!')"
      onunload="alert('谢谢!')">
<H2>onload 和 onunload 事件示例</H2>
</BODY>
</HTML>
```

这段代码的效果为：当进入页面时，显示“ 欢迎光临 ”对话框，如图 4.9 所示；退出网页（单击工具栏中的“ 前进 ”按钮跳转到另外一个页面）时，显示“ 谢谢 ”对话框，如图 4.10 所示。

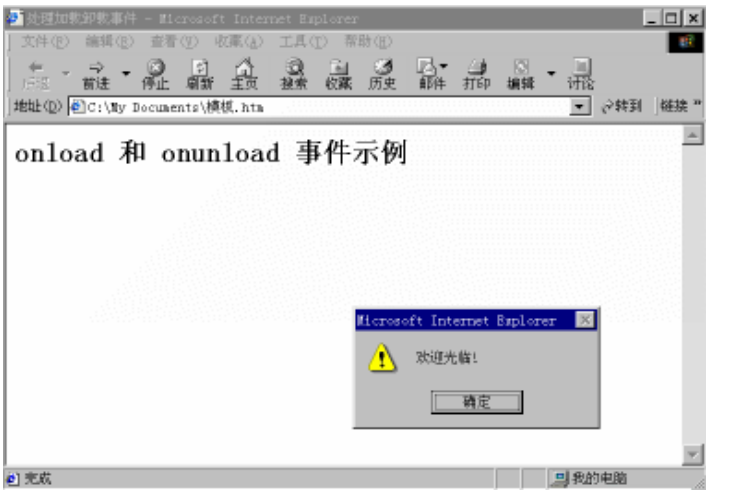


图 4.9 加载卸载事件示例

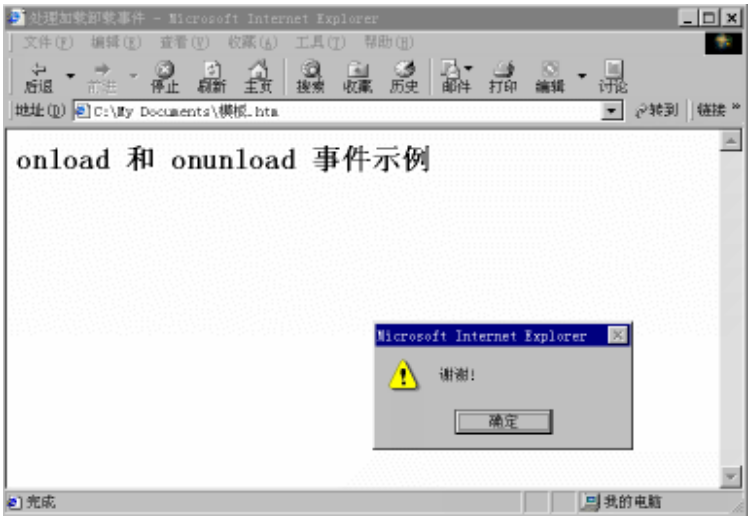


图 4.10　卸载网页时的效果

4.4　document 对象的方法

4.4.1　方法列表

表 4.2 列出了 document 对象的各种常用方法以及相应的说明。

表 4.2　document 对象的方法

方　法	说　明
close()	关闭文档的输出流。调用此方法时，在调用方法之前没有被显示到页面中的输入内容都将被显示出来。
open(mimeType)	清除当前文档并为要放置到该文档中的新数据打开一个流。该方法可以接受一个可选的参数 mimeType，以指定要写到文档中的数据的类型。该参数可以取下面的标准 mimeType 中的一种：text/html、text/plain、image/gif、image/jpeg 或 image/x-bitmap，但 IE 仅支持 text/html。
write(value1,value2...)	将用逗号分隔的参数作为字符串添加到文档中。如果参数有不是字符串，则在被添加到文档之前转换为字符串。
writeln(value1,value2...)	将用逗号分隔的参数作为字符串添加到文档中。与 write() 方法的不同之处在于：writeln() 方法在最后一个参数写入之后在文档中添加一个换行符。如果有参数不是字符串，则在被添加到文档之前转换为字符串。

说明： close() 和 open() 方法中的文档流概念是指构成所建文档的字符串序列。

4.4.2　方法示例

本示例显示了 document 对象各方法的使用，并对比了 write()方法和 writeln()方法的不

同之处，代码如下：

```
<HTML>
<HEAD>
  <TITLE>使用 document 对象的方法</TITLE>
</HEAD>
<BODY>
<H3>本示例显示了 open()、close()、writeln() 和 write() 方法的应用：</H3>
<SCRIPT Language ="JavaScript" TYPE="text/javascript">
<!--
document.open()
document.writeln("<pre>Hello")
document.writeln("World</pre>")
document.write("<pre>Hello")
document.write("World</pre>")
document.close()
//-->
</SCRIPT>
</BODY>
</HTML>
```

说明：由于在 HTML **中忽略文档中的换行符，因此必须使用预格式化文本标记符** PRE **使换行符显示出来，才能看出** write() **和** writeln() **的区别。**PRE **标记符的作用是：按照在文档中输入时的格式显示文档。**

该示例的执行效果如图 4.11 **所示。**

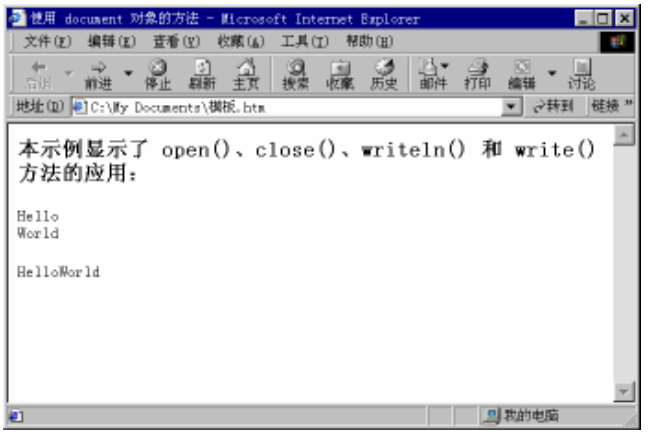


图 4.11　document **对象方法示例**