第 5 章　JavaScript脚本语言

前面介绍的 HTML 仅仅是一种标记语言，不是编程语言。标记语言设计的目的是为了组织数据，支持页面显示格式。编程语言用于作出逻辑处理，例如进行表达式条件判断，根据判断结果进行相应数据处理和动态效果。HTML 语言不能实现这一点，需要在 HTML 文件种插入一段小程序来实现，这段小程序就叫着脚本，编写脚本的语言就是脚本语言。这里将要介绍的 JavaScript 就是一种脚本语言，常用的脚本语言还有 VBScript。

本章要点包括以下内容：

- JavaScript 的介绍
- JavaScript 常量与变量的定义
- JavaScript 的表达式与运算符介绍
- JavaScript 的基本程序语句
- JavaScript 的事件介绍
- JavaScript 的对象介绍

5.1　JavaScript概述

综合来说，JavaScript 是一种基于对象和事件驱动并具有安全性能的脚本语言，它由客户端浏览器进行解析和执行。使用它的目的是与 HTML 超文本标记语言一起实现一个 Web 页面与 Web 客户交互。它通过嵌入或调入到标准的 HTML 语言中实现，它弥补了 HTML 语言的缺陷。JavaScript 是一种比较简单的编程语言，使用方法是向 Web 页面的 HTML 文件添加一个脚本，不需单独编译解释。当一个支持 JavaScript 的浏览器打开这个页面时，它会读出这个脚本并执行其指令。因此 JavaScript 使用较容易方便，运行快，适用于较简单的应用。

另外，JavaScript 可以实现很多页面效果，对键盘和鼠标事件进行监控和操作。当某些简单的处理不想在服务器端实现而加重服务器的负担时，可以使用 JavaScript 实现，例如输入框字符校验。

在后面将要介绍的 Ajax，是对 JavaScript、CSS 以及 XML 等相关技术的创新性的结合使用。Ajax 是用来实现异步处理，提高用户体现，这些都是 Web2.0 范畴。Ajax 中主要使用的就是 JavaScript 语言，所以读者在这一章认真了解 JavaScript 相关知识还是很有必要的。

总体来说，JavaScript 语言的特点包括如下：

- 是一种脚本编写语言：解释性语言（类似 Java 语言），提供简易开发过程；
- 基于对象的语言：意味着它可以运行自己已经创建的对象；
- 简单性：它基于 Java 基本语句和控制流之上的简单而紧凑的设计，它的变量类型采用弱类型；
- 安全性：JavaScript 编写的程序不允许访问本地硬盘，而且不能将数据存入到服务器上，不允许对网络文档进行修改和删除；
- 动态性：它可以直接对用户或者客户输入做出响应，无须经过 Web 服务程序；
- 跨平台性：JavaScript 是依赖浏览器本身，与操作系统无关。

5.2　编写第一个JavaScript脚本

其中在第 3 章中，读者就已经接触到了 JavaScript。当时使用 JavaScript 脚本语言来验证用户输入信息是否为空。下面在 HTML 页面程序中插入一个 JavaScript 脚本语言，以实现显示当前日期的功能，整个页面的详细代码如下：

```
<html>
<head>
<title>第一个 JavaScript</title>
<script language="javascript">
<!--
function showDate(){
   var today = new Date();              //在 JavaScript 中获取一个日期对象
   year = today.getYear();              //获取当前的年份
   month = today.getMonth();            //获取当前的月份
   date = today.getDate();              //获取当前月份的第几日
   alert("今天是"+year+"年"+month+"月"+date+"日");
}
-->
</script>
</head>

<body>
<input type="button" onClick="javascript:showDate();" value="查看今天日期">
</body>
</html>
```

代码说明：使用<script>标签表示脚本语言（可以是 JavaScript，也可以是 VBScript，或者其他）的开始，其中使用 Language 属性定义脚本语言为 JavaScript（默认为 JavaScript）。Date 为 JavaScript 保留的日期对象，可以调用其中的方法获取到当前日期的种种信息。JavaScript 采用的是弱类型，即所有类型变量都是可以通过 var 来定义的，甚至 var 字段不要都可以。本页面达到的效果为单击“查看今天日期”按钮，会弹出如图 5.1 所示的提示框。

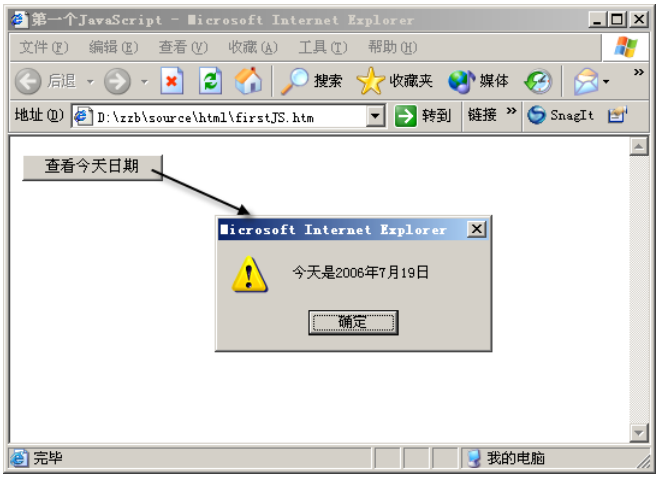

图 5.1　提示日期的 JavaScript 脚本

5.3　JavaScript基本语法

JavaScript 是一种语言，下面重点介绍这种语言的基本语法。

5.3.1　常量和变量的定义

任何一种语言都离不开常量与变量的定义，这两个概念也是最为基础和简单的，介绍 JavaScript 也首先从这里开始。

5.3.1.1　常量

JavaScript 的常量通常又称字面常量，它是不能改变的。在 JavaScript 中，常量有一下 6 种基本类型：

- 整数型常量：例如 var num=100 JavaScript 对一个常整数的定义；整数值可以使用十六进制、八进制以及十进制表示。
- 实数型常量：实数型常量是由整数部分加小数部分表示，如 12.33、168.34。可以使用科学或标准方法表示：5E7、4e5 等。
- 布尔值：布尔类型的变量只有两种值：true 和 false，主要使用它来表示一种状态，以说明操作流程。
- 字符型常量：使用单引号（ ‘ ’ ）或者双引号（ “ ” ）括起来的一个或者几个字符，例如：‘A’、“this is a book”、“12345”等。
- 空值：JavaScript 中有一个 NULL 空值，表示什么也没有，如试图引用没有定义的变量，则返回一个 NULL 值。
- 特殊字符：JavaScript 中可以使用“/”反斜杠开发来显示紧跟后面的特殊字符，例如：“/<”，将显示<特殊字符。

5.3.1.2　变量

变量是用来存放信息、获取数据的容器。对于一个变量，必须明确变量的命名、变量的类型、变量的声明以及变量的作用域：

（1）变量的命名

JavaScript 语言的变量命名同其他计算机编程语言非常相似，但是要注意以下两点：

- 必须是一个有效的变量，即以字母开头，中间可以出现数字或者下划线“_”，但是不能出现空格、“+”、“-”或者“,”等特殊字符。
- 不能使用 JavaScript 中已经出现的关键字作为变量。在 JavaScript 中有 40 多个关键字，例如 var、int、double、true 等。

（2）变量的类型

变量可以有四种类型：整数变量（例如 x=100）、字符串变量（例如 y=”this is string”）、布尔型变量（例如 xy=true）、实型变量（例如 CONST=19.5）。

（3）变量的声明

在 JavaScript 中，由于变量采用的是弱类型，不管定义的变量为何种类型，都是可以通过 var 关键字来对变量作声明。另外，变量也可以不声明而直接使用，JavaScript 是在使用到这个变量的时候再根据数据的类型来确定其变量的类型。

（4）变量的作用域

JavaScript 中有全局变量和局部变量。全局变量是定义在所有函数体之外，其作用范围是整个函数；而局部变量是定义在函数体内，只对该函数是可见的。了解过其他计算机语言的读者应该对这个概念不难理解。

5.3.2　表达式和运算符

在 JavaScript 中定义完变量之后，就可以对这些变量进行赋值、改变、计算等一系列操作。这一过程通常是通过表达式来完成的，而对表达式做的大部分工作是在做运算符处理。

5.3.2.1　算术运算符

算术运算符包括加、减、乘、除和其他数学运算，详细情况见表 5.1 所示。

表 5.1　算术运算符

算术运算符	描　　述
+	对两个数或者字符串进行加操作
-	对两个数进行减操作
*	对两个数进行乘操作
/	对两个数进行除操作
%	对数进行取模操作
++	对数进行递加1操作
--	对数进行减1操作

5.3.2.2　逻辑运算符

逻辑运算符比较两个布尔值（true 或者 false），然后返回一个布尔值，如表 5.2 所示。

表 5.2　逻辑运算符

逻辑运算符	描　　述
&&	逻辑与
||	逻辑或
!	逻辑非

5.3.2.3　比较运算符

比较运算符可以对两个数值或者表达式进行比较，并将结果以布尔类型返回，详细见表 5.3 所示。

表 5.3　比较运算符

比较运算符	描　　述
<	小于比较
>	大于比较
<=	小于等于比较
>=	大于等于比较
==	等于比较
!=	不等于比较

5.3.2.4　条件表达式

表达式的概念在前面已经介绍，它是由任何合适的常量、变量与操作符相连接而组成的式子，这个式子可以得出一个惟一的值。而条件表达式是一个特殊的表达式，它基本的语法为：

```
（条件）？A : B
```

括号中的条件返回的是一个布尔类型的值，若结果是真（true），则整个条件表达式返回的结果为 A（A 可以是确切的数值，也可以是一个数学表达式）值，否则为 B。

下面通过一个实例来演示条件表达式的具体使用方法，创建的代码如下：

```
<html>
<head>
<title>JavaScript 中的条件表达式实例</title>
</head>

<body>
<script language="javascript">
<!--
a = (10 > 8) ? "JavaScript":"VBScript";
b = (10 < 8) ? "JavaScript":"VBScript";
document.write("<center>");
document.write(a);
document.write("<br><br>");
document.write(b);
document.write("</center>");
-->
</script>
</body>
</html>
```

代码说明：该编写的 JavaScript 代码，在页面一装载的时候就会自动运行，不需通过单击按钮触发事件来调用 JavaScript 中定义的方法。这里的 document 是 JavaScript 自带的对象，代表该编写的页面，调用 write()在该页面中输出内容。

运行该页面，由于表达式“10>8”为真，所以 a 的值为“JavaScript”，而“10<8”为假，所以 b 的值为“VBScript”。运行效果如图 5.2 所示。

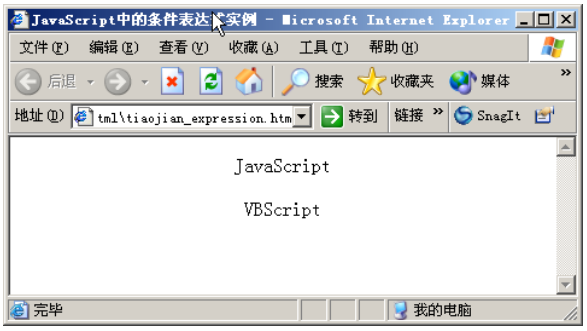

图 5.2　条件表达式

5.3.3　基本程序语句

在程序当中有时需要进行适当的判断和循环操作，下面将介绍有关 JavaScript 中的流程控制语句。如下：

5.3.3.1　If　else 语句

该程序语句的基本语法如下：

```
If(表达式条件){
   执行语句 1
} else {
   执行语句 2
```

```
}
```

代码说明：如果表达式条件成立，即返回为 true，则程序执行相应的语句 1，否则执行语句 2。下面通过实例来演示，实例代码如下：

```
<html>
<head>
<title>if else 语句实例</title>
</head>
<body>
<script language="javascript">
<!--
var index = 13;
if(index < 12)                    //判断 index 与 12 的大小关系，如果 index 小于 12
    alert("good morning");
else if(index < 18)               //如果 index 小于 18
    alert("good afternoon");
else                              //否则
    alert("good evening");
-->
</script>
</body>
</html>
```

代码说明：由于“index < 18”表达式条件为真，所以会弹出“good afternoon”的提示框。运行效果如图 5.3 所示。

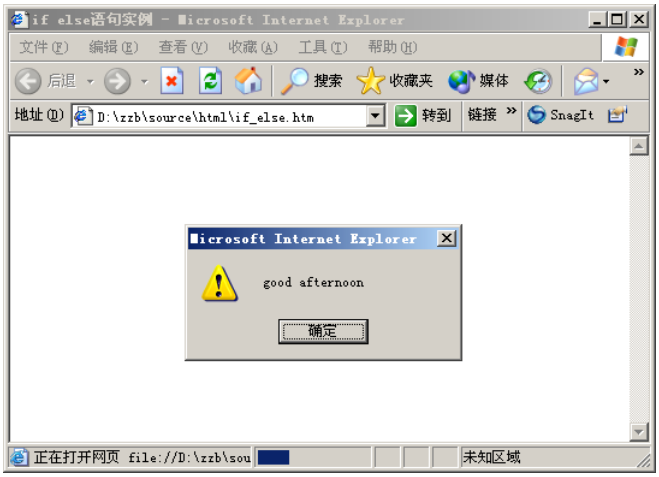

图 5.3　if else 语句

5.3.3.2　For 语句

For 语句的基本语法如下：

```
For(初识部分;条件部分;更新部分)
{
  执行语句
}
```

代码说明：初识部分一般设定循环的初识值，条件部分进行条件判断，如果条件成立，则进行初识值的更新以及进行相应语句的操作，然后再进入下一个循环过程。

示范的实例代码如下：

```
<html>
<head>
<title>for 循环语句</title>
</head>
<body>
<script language="javascript">
<!--
var end = 5;
 for(i=0;i<end;i++)                    //循环执行 end 次
 {
     document.write("这是第 "+(i+1)+" 次循环操作<br>");
  }
-->
</script>
</body>
</html>
```

代码说明：end 变量定义了循环的总次数，使用 for 语句依次在页面中输出第几次循环的提示语，该页面的运行效果如图 5.4 所示。

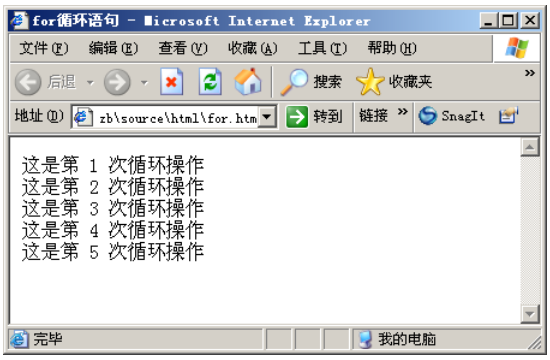

图 5.4　for 语句

5.3.3.3　函数

函数为程序设计人员提供了一个非常方便的操作方法。通常在进行一个复杂的程序设计时，总是根据所需要完成的功能，将程序划分成一些相对独立的部分。每个部分编写成一个函数，类似于 Java 语句中类概念和思想。从而，可以使得程序充分独立、任务单一、清晰、易懂、易读以及易于代码的维护和扩展。基本语法：

```
Function  函数名(参数 1,参数 2,…)
{
   函数执行部分
   Return  表达式
}
```

代码说明：函数的定义是通过 function 关键字来声明的，并且函数可以定义传入的多个参数。函数执行部分一般进行相应逻辑操作和输出执行结果，return 关键字返回表达式运行结果。

下面同样通过一个实例来演示该语句的具体使用方法，实例代码如下：

```
<html>
<head>
<title>函数使用实例</title>
<script language="javascript">
```

```
<!--
   function f(y)
   {
      var x = y*y;
       return x;
   }
-->
</script>
</head>

<body>
<script language="javascript">
<!--
   var y = 3;
   var x = f(y);
   alert("传入参数 3,函数 f()运行的结果为: "+x);
-->
</script>
</body>
</html>
```

代码说明：f()函数对传入的参数进行了平方操作，然后通过 return 返回运算结果。该页面运行的效果如图 5.5 所示。

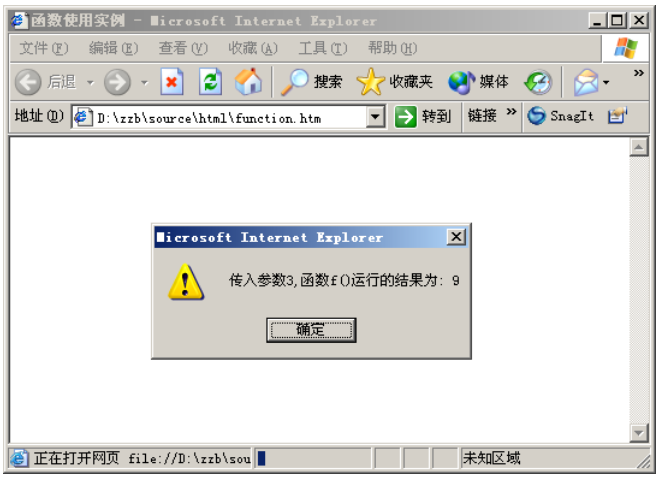

图 5.5　函数使用

5.3.4　JavaScript的事件

JavaScript 是一个基于面向对象（Object－based）的语言，而基于对象的基本特征，就是采用事件驱动（event－driven）。通过鼠标或者热键的动作称为事件（event），而由鼠标或者热键引发的一连串程序的动作，称之为事件驱动（Event driver），对事件进行处理的程序或者函数称之为事件处理程序（Event Handler）。

5.3.4.1　使用事件的方法

至此，读者应该对事件并不陌生了，因为在第 3 章以及本章的开头实例中都使用到一个最为常见的 onclick 事件。当用户单击鼠标按钮时，产生一个 onclick 事件，通过 onclick 指定的事件处理程序或者代

码将被调用执行。

具体调用方法请查看前面的实例。

5.3.4.2　常用事件介绍

在 JavaScript 中，常用的事件介绍如表 5.4 所示。它们的使用方法都类似于 onclick 事件，这里就不一一讲解了。

表 5.4　常用事件

事　　件	描　　述
Onclick	鼠标单击时触发事件
Onchange	文本框的内容改变时发生的事件
Onselect	文本框的内容被选中时发生的事件
Onfocus	光标落在文本框中时发生的事件
Onload	当前网页刚被打开时候发生的事件
Onunload	当前页面被关闭时候发生的事件

5.3.5　JavaScript的对象

JavaScript 语言是基于对象的，将复杂对象统一起来，从而可以形成一个非常强大的对象系统。下面重点介绍浏览器中自带的对象，从而 JavaScript 语句中可以直接调用对象的方法。

浏览器对象就是网页和浏览器本身各种实体元素在 JavaScript 程序中的体现。这样的浏览器对象主要包括：

- Navigator 对象：管理着当前使用浏览器的版本号、运行的平台以及浏览器使用的语言等信息。
- Window 对象：处于整个从属表的最顶级位置，每一个这样的对象代表一个浏览器窗口。
- Location 对象：含有当前网页的 URL 地址。
- Document 对象：含有当前页面的各种特性，例如标题、背景以及使用的语言等。
- History 对象：含有以前访问过的网页的 URL 地址。

使用浏览器的内部对象系统，可以实现与 HTML 文档进行交互。它的作用是将相关元素组织包装起来，提供给程序设计人员使用，从而减轻编程人员的劳动，提高设计 Web 页面的能力。当然编程人员也可以自行编写对象，但是由于本书重点不是 JavaScript，所以这里就不作重点介绍。下面一一介绍这些常用的浏览器对象。

5.3.5.1　Navigator 对象

该对象包括了整个浏览器的环境信息，其中包括的常用属性如下：

- AppName：保存浏览器名称。例如，使用 Navigator 浏览器时，appName 值为“NetScape”；当使用的是 Iternet Explorer 浏览器时，appName 值为“MSIE”。
- AppVersion：反映浏览器的版本号。
- AppCodeName：反映用字符串表示的当前浏览器的代码名称。对于 Navigator 的所有版本，这个值都是“Mozilla”。

下面通过一个简单实例演示该对象的属性调用，详细代码如下：

```
<html>
<head><title>Navigator 对象</title></head>
<body>
<script language="javascript">
<!--
```

```
  var appname = navigator.appName;
  var appversion = navigator.appVersion;
  document.write("你使用的是"+appname+"浏览器<br>版本号为:"+appversion);
-->
</script>
</body>
</html>
```

代码说明：调用 navigator 对象中的 appName 和 appVersion 属性获取到浏览器的名称和版本号。然后再通过 document 对象中的 write 方法将这些信息输出在页面中。注意对象以及对象属性的大小写。

该页面的运行效果如图 5.6 所示。

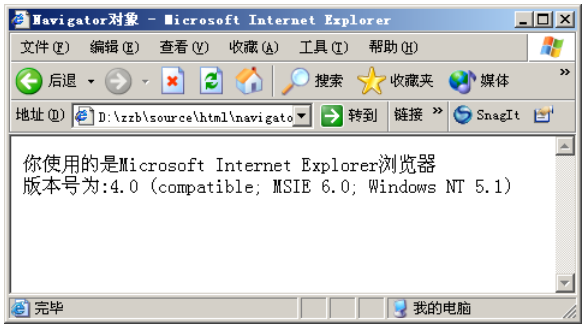

图 5.6　navigator 对象调用

5.3.5.2　Window 对象

窗口对象 window 包括了很多有用的属性、方法和事件驱动程序，编程人员可以利用这些对象控制浏览器窗口显示的各个方面，如对话框、框架等。下面列出了一些常用的 Window 对象方法：

- Open(URL,windowName,parameterList)方法：open 方法创建一个新的浏览器窗口，并在新窗口中载入一个指定的 URL 地址。
- Close()方法：close 方法关闭一个浏览器窗口。
- Alert()方法：弹出一个消息框。
- Confirm()方法：弹出一个确认框。
- Prompt()方法：弹出一个提示框。

下面演示一下 Window 对象的使用实例，详细代码如下：

```
<html>
<head><title>window 对象实例<title>
<script language="javascript">
<!--
function createNew(){
window.open("8-1.html","new_window","height=100,width=300,top=100,left=100,toolbar=no,menubar=no,scrollbars=no,resizable=no,location=no,status=no");
}
-->
</script>
</head>
<body>
<input type="button" onclick="createNew();" value='弹出新窗口'/>
</body>
</html>
```

代码说明：该页面编写的 JavaScript 语句中使用 window 对象来打开一个名为“8-1.htm”新页面。其中“new_window”是新窗口的名称，另外定义了窗口的高、宽、距离窗口顶端和左端宽度。“toolbar”属性定义新窗口是否带工具栏；“menubar”属性定义是否含有菜单栏；“scrollbars”属性指定是否有滚动栏；“resizable”属性规定窗口大小是否可以改变。

该页面的运行效果如图 5.7 所示。

图 5.7　弹出新窗口页面

5.3.5.3　Location 对象

Location 对象是当前网页的 URL 地址，可以使用 Location 对象来让浏览器打开给定的页面。示范实例代码如下所示：

```
<html>
<head>
<title>Location 对象</title>
</head>
<body>
<form>
<input type="button" value="单击这里" onClick="window.location.href='new.htm';">
</form>
</body>
</html>
```

代码说明：该段 JavaScript 代码直接写入到<input>标签中，由 onclick 事件来触发。JavaScript 就是这么灵活。该段代码的演示效果如图 5.8 和 5.9 所示。

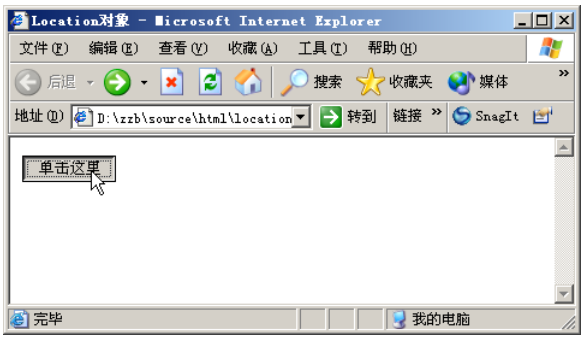

图 5.8　Location 对象

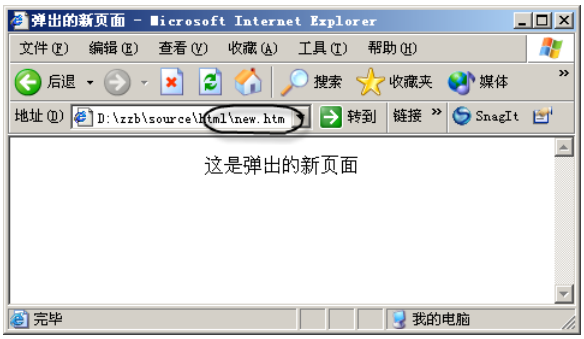

图 5.9　弹出的新页面

5.3.5.4　document 对象

在 document 中主要有：Links、Anchor 以及 form 等三个主要的对象：

- Anchor 锚对象：Anchor 对象指的是标记在 HTML 源代码中存在时产生的对象，它包含着文档中所有的 anchor 信息。
- 链接 links 对象：Link 对象指的是用标记链接一个超文本或者超媒体的元素作为一个特定的 URL。
- 窗体 form 对象：窗体对象是文档对象的一个元素，它含有多种格式的对象存储信息，使用它可以在 JavaScript 脚本中编写程序进行文本输入，并可以用来动态改变文档的行为。通过 document.forms[]数组使得在同一个页面上可以有多个相同的窗体，使用 forms[]数组要比使用窗体名字方便得多。

下面通过一个实例来使用 form 对象，创建的代码如下：

```
<html>
<head>
<title>document 对象</title>
</head>
<body>
<form>
<input type="text" onChange="document.form1.elements[0].value = this.value;">
</form>
<form name="form1">
<input type="text" onChange="document.forms[0].elements[0].value = this.value;">
</form>
</body>
</html>
```

代码说明：在第一个表单中的文本框中输入文字，第二个表单中的文本框会自动填写上相同的值，相反也是一样的。其中的 forms 是一个数组对象，可以通过索引来调用页面中的各个表单对象，另外页面可以直接通过表单的名称来调用。该页面的运行效果如图 5.10 所示。

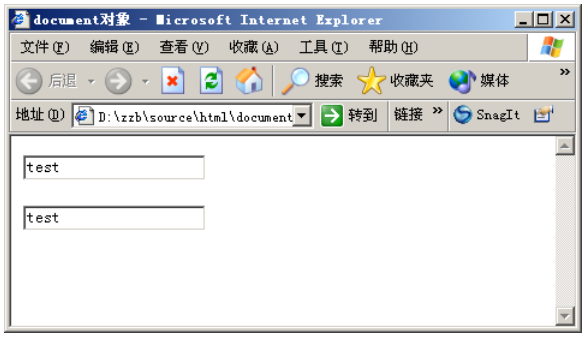

图 5.10　document 对象

5.3.5.5　History 对象

History 对象中含有以前访问过的网页的 URL 地址，下面的实例将演示页面的前进和后退功能，详细代码如下：

```
<html>
<head>
<title>history 对象</title>
</head>
<body>
<input type="button" value="后退" onclick="history.go(-1);">
<input type="button" value="前进" onclick="history.go(1);">
```

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

代码说明：给 history 对象中的 go()方法传递－1 时，表示后退操作；当传递 1 参数值，表示前进操作。该页面运行效果如图 5.11 所示。

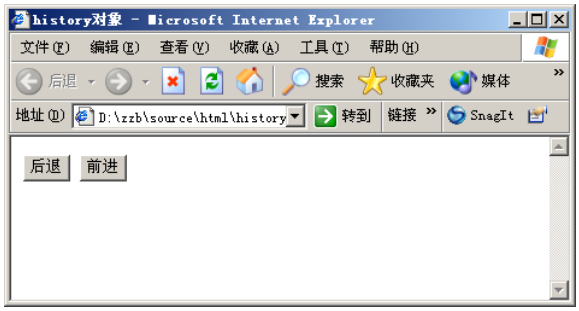

图 5.11　history 对象

至此，本章已经对 JavaScript 的基本使用方法介绍完毕，但是 JavaScript 的其他使用方面还很丰富。由于本书的主题不是 JavaScript，这里介绍 JavaScript 是为了下面将要介绍的 Ajax。如果读者对 JavaScript 其他方面感兴趣，请自行查看相关资料。

5.4　本章小结

JavaScript 是浏览器脚本语言，是由客户端的浏览器来解析和执行的，执行部分的动态效果，可以减轻服务器端的部分负担，例如表单数据的校验。在本书的高级篇幅中，将要介绍的 Ajax 就是基于 JavaScript 语言来实现的，目标是为了提高用户体验，所以本章的 JavaScript 学习是很有必要的。