目　录

第1章 JavaScript 基 础

主要内容

- 关 于 JavaScript
- 了 解 JavaScript
- World Wide Web
- Web 应用程序
- JavaScript 与 VBScript

本章导读

JavaScript 作为一种新的 Web 技术，自出现以来就获得了各方面的广泛支持。本章介绍了 JavaScript 的基础知识以及与其它脚本语言的区别。

1.1 关 于 JavaScript

JavaScript 是一种新的 Web 技术。JavaScript 最初的名字是 LiveScript，它是 Netscape 开发出的一种脚本语言，其目的是为了扩展基本的 HTML 的功能，用于代替复杂的 CGI 程序来处理 Web 页表单信息，为 Web 页增加动态效果。当 Java 出现以后，Netscape 和 Sun 一起开发了一种新的脚本语言，它的语法和 Java 非常的类似，所以它最后被命名为 JavaScript。

在 JavaScript 出现以前，在 Web 页中需要进行的所有处理都必须传回服务器，由服务器进行集中处理。服务器处理完毕后，在将处理结果通过网络传回客户端的浏览器中供用户查看。即使是最简单的验证用户在文本框中输入数据的有效性，比如通过判断输入字符串是否包含“@”符号来判断用户输入的 E_mail 地址是否有效，都必须由服务器来完成。在这种方式下，当 Web 访问量增加时，网络和服务器的负担都会增加。这一时期的客户/服务器结构并不是真正意义上的客户/服务器结构。人们期待一种新的技术来实现真正的客户/服务器结构，即在客户端也可以进行处理，从而减轻服务器的负担，加快网络的传输速度。JavaScript 正是在这种背景之下产生的。

JavaScript 至 1995 年诞生以来，已经取得了广泛的支持，他们包括 Apple、Borland、Sybase、Informix、Oracle、Digital、HP 和 IBM 等。不仅仅是在浏览器中得到越来越多的支持，在其它的各种应用程序中也得到应用。新的 Windows 操作系统中也可以使用脚本来制订需要完成的任务。

有经验的 Web 页作者都知道 Java 的小应用程序也可以实现客户端的逻辑处理能力。但作为一种强类型的程序设计语言，Java 并不是制作 Web 页的最佳选择。因为使用 Java 需要 Web 页作者有较高的编程能力，而这对众多 Web 页作者来说是件难事。Web 作者愿意使用更简单的方法来实现表单的处理。

Microsoft 也意识到了 Web 脚本的重要性。作为软件界的领头羊，Microsoft 自然不甘在 Web 脚本的竞争中落后。由于得不到 Netscape 在技术上的许可，Microsoft 开发了自己的脚本语言——JScript，并在 Microsoft 自己的浏览器 Microsoft Internet Explorer 3.0 以及更高版本中对其提供支持。由于 Microsoft 在浏览器市场中的优势，JScript 很快得到广泛支持和应用。JScript 1.0 只是很粗糙地和 JavaScript 1.1 兼容，Netscape 在其浏览器 Navigator 3.0 及其以后的版本中也对 JScript 提供了支持。随着 JavaScript 版本的增多和浏览器平台的不同，让众多的 Web 页作者感到难以取舍，也增加了额外的工作量。

鉴于脚本语言开发商之间的竞争给 Web 页作者带来的麻烦，Microsoft、Netscape 和其它脚本语言商决定成立一个国际组织，并将其命名为 ECMA，该组织专门从事脚本语言标准的制订。ECMA 制订的脚本语言标准被称为 ECMAScript，所有开发商的脚本语言都支持这一标准。尽管有 ECMA 标准的存在，Netscape 和 Microsoft 都有其各自的脚本语言——JavaScript 和 JScript，这两种语言都对 ECMA 标准进行了扩展。

Microsoft 除了 JScript 之外，还有 VBScript 也是一种脚本语言。VBScript 实际上是 Visual Basic 程序设计语言的一个子集，使得众多的 VB 程序设计员很容易编写自己的 Web 应用

程序。Netscape 并没有对 VBScript 提供支持，所以使用 VBScript 的还仅是 Microsoft Internet Explorer 用户。

即使有 VBScript 的竞争，JavaScript 还是成为了标准的 Web 脚本语言。在人们的印象中，JavaScript 只是用来编写客户端的 Web 应用程序。Netscape 为用户提供了服务器端的脚本语言 Netscape Server_Side JavaScript（SSJS），可以在服务器端编写需要的 Web 应用程序，不过使用 SSJS 需要 Netscape Server 3 的支持。Microsoft 也有自己的服务器端脚本编程语言 Active Server Pages（ASP）——需要 JScript 引擎的支持。

1.2 了 解 JavaScript

学习 JavaScript 这样的新工具可以说是一种挑战，因为很难理解它是如何使用以及能用于哪些方面。学习 JavaScript 并不是一件非常困难的事，我们先从下面的 10 个方面了解一下 JavaScript 的特点。

1.2.1 JavaScript 是被嵌入到 HTML 中的

JavaScript 的最大特点便是和 HTML 的结合。在客户端的应用中，很难将 JavaScript 程序和 HTML 文档分开。JavaScript 代码总是和 HTML 一起使用的，JavaScript 的各种对象都有各自的 HTML 标记。当 HTML 文档在浏览器中被打开时，JavaScript 代码才被执行。JavaScript 代码使用 HTML 标记<script>……</script>嵌入到 HTML 文档中。JavaScript 扩展了标准的 HTML，为 HTML 标记增加了事件，通过事件驱动来执行 JavaScript 代码。在服务器端，JavaScript 代码可以作为单独的文件存在，但也必须通过在 HTML 文档中调用才能起作用。

下面的程序清单 1.1 中的例子说明了 JavaScript 代码是如何嵌入到 HTML 文档中的。

程序清单 1.1　在 HTML 文档中嵌入 JavaScript 代码

```
<html>
<head>
<title>在 HTML 文档中嵌入 JavaScript 代码</title>
<script language="javascript">
    <!--
    window.defaultStatus="使用 HTML 标记嵌入 JavaScript 代码"
    function getnews()
     {
      document.form1.textbox.value = "在 HTML 文档中使用 JavaScript 代码"
     }
     //-->
</script>
</head>
```

```
<body>
<center>
<h1>使用 JavaScript</h1>
<hr>
<form name="form1">
<input type=text name="textbox" size=40 value="单击按钮查看信息">
<br><br>
<input type=button value = "查看信息" onclick = "getnews()">
</form>
</center>
</body>
</html>
```

本例仅说明如何在 HTML 文档中嵌入 JavaScript 代码，在后面的章节中将详细介绍 JavaScript 的使用。图 1-1 显示了本例在浏览器中打开的实际效果。

图1-1 在 HTML 文档中嵌入 JavaScript 代码

1.2.2 JavaScript 需要环境的支持

JavaScript 是作为一种语言而不是工具出现的。工具是可以不依赖环境而单独使用，但 JavaScript 的运行需要环境的支持。运行 JavaScript 的浏览器，如 Microsoft Internet Explorer 和 Netscape Navigator，或者服务器端引擎是一个解释引擎。JavaScript 只有在这些解释引擎的支持下才能发挥强大作用。如果使用的浏览器不支持 JavaScript，那么嵌套在 HTML 文档中的 JavaScript 代码将会被忽略。最严重的结果是 JavaScript 代码被不支持的浏览器当作文本原样显示在浏览器中。

由于环境因素的存在，在编写 JavaScript 应用程序时必须考虑我们在何时何地使用它以及浏览器是否支持。如果浏览器不支持我们又该如何实现？可以使用非 JavaScript 方法来解决吗？所有的问题都需要我们在编写 JavaScript 应用程序之前进行考虑。

一个有力的说明是，帧在刚发布的时候，虽然是一种新的、革命性的技术，但这一技

术却只有 Netscape Navigator 2.0 支持。这让 Web 作者难以决定什么时候使用帧，而当浏览器不支持时也不知道该如何进行处理。后来 Microsoft Internet Explorer 3.0 开始支持帧，在 HTML 4 中帧已成为了标准。

现在 Web 作者不用再对 JavaScript 的环境依赖性做过多的考虑，JavaScript 已获得了众多浏览器的支持。Netscape Navigator 2.x、Microsoft Internet Explorer 3.x、Opera 3.x 和 HotJava 3.x 等主要浏览器以及它们的更高版本都能支持 JavaScript。一个数字统计表明，大约有 95%的浏览器现在都对 JavaScript 提供支持，这对 JavaScript 使用者来说是一个福音。

1.2.3 JavaScript 是解释执行的

和大多数脚本语言一样，JavaScript 在浏览器中是解释执行的。应用程序的执行通常有解释和编译两种方式。编译是将程序源代码翻译成可执行的二进制代码文件，如.EXE 文件，而解释则是翻译一句就执行一句。JavaScript 代码并不被编译为二进制代码文件，而是作为 HTML 文件的一部分。由浏览器解释执行 JavaScript 代码的缺点是代码的执行需要花更长的时间；优点则是 Web 页的维护和更新更加方便。管理员可以直接打开 HTML 文件来编辑修改 JavaScript 代码，而用户则可通过浏览器立即看到新的结果。

值得注意的是，如果是在 Netscape 的服务器端使用 JavaScript，需要用户将所有的 JavaScript 代码和 HTML 文件编译成字节代码，并存储在.web 文件中。

1.2.4 JavaScript 是一种弱类型语言

JavaScript 与 Java 和 C++等强类型语言不同，强类型语言要求用户在程序中使用一个变量之前必须先进行声明。JavaScript 则显得非常灵活，在使用一个变量时，可以先进行声明，也可以不那样做。下面的例子说明了在 JavaScript 中使用变量的灵活性。

```
<html>
<head>
<title>JavaScript 是弱类型的</title>
<Script Language="JavaScript">
    <!--
    var myVar   //声明一个变量
    myVar="JavaScript 是弱类型"   //为变量赋值
    alert(myVar)   //使用消息框显示变量值
    myVar=3.1415926   //为变量赋不同类型的值
    alert(myVar)
    a="使用未声明变量"   //使用未声明的变量
    alert(a)
     //-->
</SCRIPT>
</head>
```

</HTML>

1.2.5 JavaScript 以对象为基础

JavaScript 中使用了面向对象程序设计（OOP）的方法。在后面的章节中，我们将看到 JavaScript 的各个自定义对象，如 Window、Document 等对象。

对象实际上是封装了的数据（属性）和行为（方法）的集合。如果你使用过 Java、Delphi 或 Visual Basic，则学习 JavaScript 将是一件轻松的事。JavaScript 的对象都是实例化了的，只可以使用而不能创建继承于这些对象的新的子类。

1.2.6 JavaScript 通过事件驱动执行

我们在后面将学习到 JavaScript 代码是如何使用事件来驱动执行的。HTML 文档中的许多 JavaScript 代码都是通过事件驱动的。JavaScript 本身支持事件。HTML 对象，如按钮和文本框，增加了对事件的支持。如果你学习过 Java 或 Visual Basic 等面向对象程序设计，则你一定知道事件驱动。如果你仅学习过像 C 语言等面向过程的程序设计，也不用担心，在后面我们详细讲解 JavaScript 的事件驱动。

1.2.7 JavaScript 不是 Java

当你在网上冲浪的时候，你可能会看到许多和 JavaScript 相关的 Web 站点在讨论相同主题：JavaScript 不是 Java。我们在前面已经提到过 JavaScript 和 Java 是两个不同公司的产品（JavaScript 是 Netscape 公司的，Java 是 Sun 公司的），名称上的相似只是出于市场的原因。在后面的学习过程中，你会体会到 JavaScript 和 Java 更多的不同。但我们可以从几个方面简单了解一下 JavaScript 和 Java 的区别：

- JavaScript 和 HTML 文档是紧密集成的。JavaScript 代码使用<SCRIPT>标签完全嵌入 HTML 文档中。Java 编写的 applet（Java 小应用程序）程序代码存放在一个单独的文件中，在 HTML 文档中使用<applet>标签建立 Java 小应用程序和 HTML 文档的联系;
- Java 比 JavaScript 更像一个完整的程序设计语言。Java 是强类型的、真正的面向对象、有自己的编译器，主要用于编写完整的应用程序和应用于 Web 页中的 Java 小应用程序。JavaScript 主要用于编写 Web 页脚本。

从语言本身的角度讲，JavaScript 的语法非常像 Java，所以两者在学习上有相通之处。

1.2.8 JavaScript 的作用是多方面的

JavaScript 在 Web 中的作用是多方面的，可从下面的几个方面得到体现：

- 可以为 Web 页增加特殊效果、动画或标志;
- 可以在客户端独立完成数据的验证;
- 用于建立客户/服务器应用程序;

- 开发客户端应用程序;
- 将 HTML 对象、Java 小应用程序、ActiveX 控件和 Netscape 插件组合一起使 Web 网页发挥强大作用;
- 扩展 Web 服务器功能;
- 在不使用 CGI 程序的情况下开发客户端数据库应用程序。

1.2.9 JavaScript 是不断发展的

JavaScript 不是一成不变的，它不断得到更新和加强，以满足不断增多的需要。到目前为止，JavaScript 已经有了 6 个版本：JavaScript 1.0 到 5.5。JScript 从 1.0 到 5.5 也经过了多个版本。所以在开发 JavaScript 应用程序过程中，不仅需要考虑浏览器是否支持 JavaScript，还需要考虑 JavaScript（或 JScript）版本之间的兼容性。

如果要随时了解 JavaScript 的发展变化，你可以访问 Netscape 的网站http://developer.netscape.com 或 http://www.mozilla.com 。Microsoft 在其网站 http://msdn.microsoft.com/scripting 中为用户提供 JScript 的最新信息。

1.2.10 JavaScript 的应用非常广泛

作为一种脚本语言，JavaScript 的应用是非常广泛的。也许许多人都会认为 JavaScript 是用来编写客户端的 Web 应用程序（这也是本书的重点），正如我们在前面提到过的一样，JavaScript 还可以用在 Netscape Enterprise 服务器和 Microsoft 的 ASP 中编写服务器端的应用程序。在一些 Web 开发工具中，如 Borland 公司的 IntraBuilder 和 Macromedia 公司的 Dreamweaver（2.0 及更高版本），也将 JavaScript 作为脚本开发语言。

另外，最新的 Windows 操作系统（Windows 98、Windows 2000）也支持 JavaScript。通过使用 Windows Script Host（WSH），你可以编写 JavaScript 程序来自动完成需要执行的任务。所以，不要老是想着 JavaScript 只能编写客户端 Web 应用程序。

1.3 World Wide Web

在过去的 6 年中 Web 技术不断的发展，使 Internet 世界已经发生了巨大的变化。最初的 Web 页只不过是简单文字，而现在 Web 已经是一个丰富多彩的世界了。在我们进一步学习如何编写 Web 应用程序前，先让我们了解一下 Web 的发展过程。Web 的发展大致可以分成四个阶段：

- **第一个阶段**：基于字符的超文本:

第一个 Web 页出现于 1989 年，由于条件的限制，人们在使用计算机访问 Web 时根本没有更好的方法来显示和处理图形。所以人们只能选择更容易的方式来通过 Web 共享信息——使用文本。在那时的 Web 页中，人们使用高亮度的文本来代表超级链接，超级链接可以让人们很容易跳转到相关的 Web 页中去。随着 Web 技术的进步，超文本技术已发生了革命性的变化。

- **第二阶段**：基于图形的 HTML 文档：

1993 年出现了第一个支持图形的浏览器 NCSA Mosic。Mosic 是一群大学生和 Netscape 的奠基人 MarcAndreesen 共同为国际超级计算机应用中心（National Center of Supercomputing Applications，NCSA）开发的 Web 浏览器。此时 Web 的概念已经在各个科研机构中得到广泛应用。图形浏览器的出现，为 Internet 开辟了新的广阔天地，使得 Web 的访问更加容易、更加具有趣味性。在同一时期图形环境很快超过基于字符的桌面系统并流行起来。

图形桌面系统加上图形 Web 浏览器，这正是人们期盼以久的梦幻组合。在仅仅几个月的时间内，在计算机界掀起一种狂热的竞争：计算机公司、机构开始为 Web 提供内容或是为客户提供 Web 服务。

这时的 Web 仍然是静态的，Web 的内容包括文本、图形，有些还包括一些声音或视频文件。不过这时的声音或视频文件需要单独下载，然后使用外部应用程序来进行播放。

- **第三阶段**：Dynamic HTML 文档：

在 Web 发展的第一和第二阶段中，Web 页是使用 HTML 文本编辑器来编写，然后放在 Web 服务器中供人们访问。放置于 Web 服务器上的 Web 页在大多数情况下都是不会改变的，除非 Web 作者进行修改。这些保持不变的 Web 页一般能满足大多数用户的要求，而对一些特别的用户来说这还是不够的。比如一个为用户提供产品目录信息的 Web 站点，每个用户需要查看的内容可能会各不相同，如果使用长期保持不变的 Web 页显然不能满足用户要求。

为了实现动态产生 Web 页，Web 页作者开始使用公用网关接口（CGI）来编写服务器端的 Web 页应用程序。CGI 程序使得用户和浏览器可以进行低级的交互行为。

- **第四阶段**：Active HTML 文档：

从 1995 年开始，人们开始在 Netscape Navigator 中使用插件和 Java。这一阶段最大的特点就是客户端——浏览器的功能得到加强，不再单独依靠服务器来运行应用程序和处理用户信息。JavaScript、Java、ActiveX 和其它客户端扩展功能的应用，使得客户端对服务器的依赖性减小，使浏览器和服务器成为真正的客户/服务器结构。浏览器成为一个功能强大的可以运行 Web 应用程序的操作环境。

1.4 Web 应用程序结构

以 Web 作为应用程序开发环境正逐渐成为一种趋势。随着 Java、JavaScript、ActiveX 和其它技术的出现，越来越多的人将兴趣集中到开发 Web 应用程序上。使用 Web 作为应用程序开发环境听起来有点让人感到迷惑，因为 Web 具有分布性的特点，所以 Web 应用程序的构成是多方面的，需要多种技术的支持。

在基于局域网的客户/服务器机构中，非常典型的应用可能就是基于客户端的数据库网络应用程序。也许你曾经使用过一些单独的工具开发过客户端应用程序，如使用 Visual Basic 或 Delphi。而在服务器端的典型开发和维护工具恐怕要数 SQL Server。

图 1-2 显示了 Web 应用程序的各个组成部分。Web 应用程序结构可以看成是各种 Web

技术的集成。开发 Web 应用程序使用的每一种技术都具有一定的局限性，应用范围相当狭小，但它们在一起使用，则可以提供有效的方法来解决 Internet 或 Intranet 应用程序的开发问题。

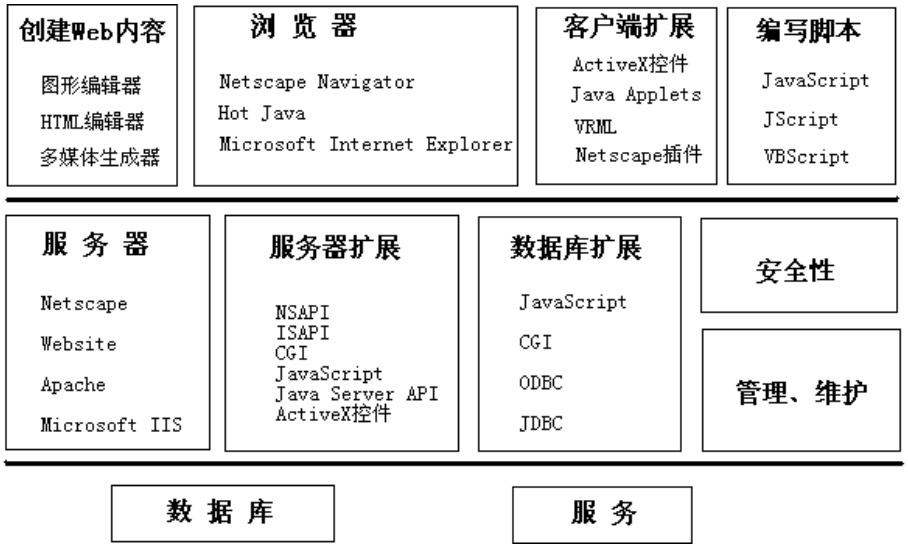


图1-2 Web 应用程序结构

1.4.1 客户端 Web 应用程序

客户端 Web 应用程序可以划分成以下四个部分：

- 浏览器;
- HTML（Hypertext Markup Language，超文本标记语言）；
- 客户端扩展（Java Applets、ActiveX 和 Netscape 插件）；
- 脚本编写语言（JavaScript、JScript 和 VBScript）。

图 1-3 显示了客户端的各种 Web 技术是如何一起工作的。下面我们将分别介绍客户端的各种技术。

1. 浏 览 器

浏览器毫无疑问是 Web 应用程序中最重要的组成部分。用户访问 Web 页，Web 应用程序的运行，都必须在浏览器中进行。浏览器的技术相对简单一些，只需要能够正确地从 Web 服务器下载 Web 页，并将其显示在浏览中。浏览器并没有一个同一的标准，当前最常用的浏览器是 Netscape Navigator 和 Microsoft Internet Explorer。不同的浏览器对各种 Web 技术有不同程度的支持，所以在不同的环境中选择浏览器对开发 Web 应用程序起着关键作用。

如果开发的是 Intranet（内网）的 Web 应用程序，那你可以确定客户端的用户使用一个标准的浏览，如使用 Netscape Navigator 或 Microsoft Internet Explorer。在确定了浏览器的前提下，Web 应用程序的开发将相对简单。而当我们将开发的 Web 应用程序发布到 Internet 上，由于客户端用户使用的浏览器是多种多样，Web 应用程序的开发将变得复杂

起来。表 1-1 列出了各种浏览器对各种 Web 技术的支持情况。

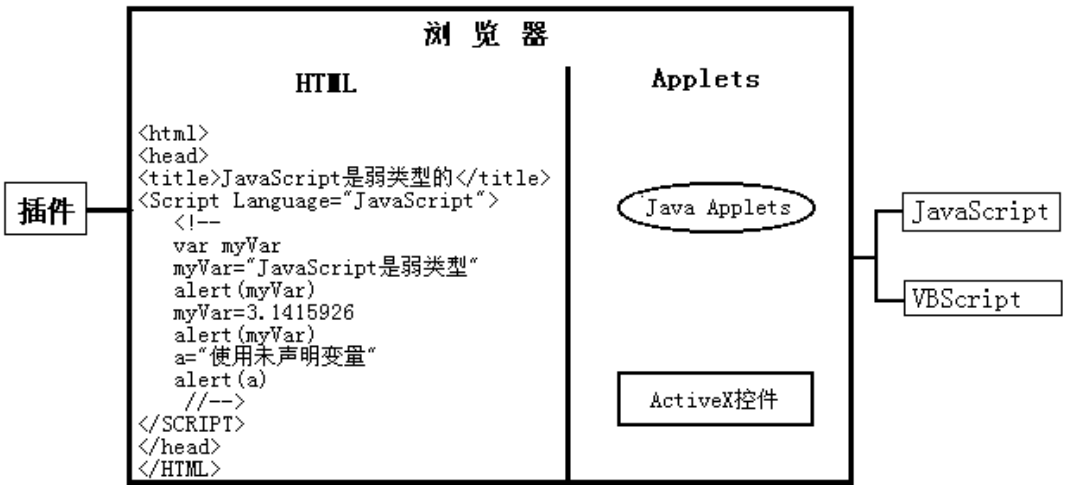


图1-3 客户端 Web 技术

表1-1

	Netscape Navigator	Microsoft Internet Explorer
JavaScript	2.0+	3.0+
VBScript	不支持	3.0+
Java	2.0+	3.0+
ActiveX 控件	3.0+插件	3.0+
Netscape 插件	2.0+	3.0+

2. HTML

HTML 也是进行 Web 应用程序开发的主要技术之一。HTML 是一种标记语言，严格地讲 HTML 并不能称之为语言。HTML 是使用各种预先定义的标记对文本进行标注，被标注的文本相当于进行了格式化。HTML 的特点就是简单，正是这种简单的特性使它更易于使用。

被标注的 HTML 文档最终在浏览器中显示出实际的效果。如果仅使用 HTML，我们可以创建静态的 Web 文档，如返回服务器端的查询结果、使用反馈表单或者显示 JavaScript 应用程序结果等。Web 应用程序的开发是离不开 HTML 的，比如 JavaScript 应用程序便需要使用 HTML 标签嵌入到 HTML 文档中才能起作用。

3. 客户端扩展

客户端扩展功能是扩展浏览器的功能，随着活动 Web 页功能的不断地加强，仅仅靠加强浏览器的功能还是不够的，所以需要对浏览器的功能进行扩展。客户端的扩展都是在浏览器中使用第三方的附加软件来加强浏览器的功能。第三方附加软件需要和浏览器一起工作，但并没有和浏览器绑定到一起。

现在出现的第三方客户端扩展功能在功能上具有相似之处，但仍然有各自的特点。主

要的客户端扩展功能有：

- Java applets
- ActiveX 控件
- Netscape 插件

Java 一出现，便迅速得到了广泛的应用。如果你从未听说过使用 Java 来编写程序，那就是真正的孤陋寡闻了。Java 是 SUN 公司开发的具有独立平台的程序设计语言。Java 吸引众多编程人员的一个最重要的特点就是使用 Java 编写的应用程序可以在不同的平台上运行。在浏览器环境中运行的 Java 应用程序我们称之为 Applets（小应用程序）。

Java 小应用程序通过使用 HTML 标签<applet>在 HTML 文档中进行引用。当用户在浏览器中引用了 Java 小应用程序的 Web 页时，Java 小应用程序被下载到用户使用的客户机中。Java 小应用程序是以字节代码的形式被下载到客户机中的，如果浏览器支持 Java，浏览器将对这些字节代码进行翻译并执行它们。如果浏览器不支持 Java，Java 小应用程序将会被忽略，用户将看不到 Java 小应用程序的效果。

ActiveX 控件最早的名称是 OCX，它是 Microsoft 在技术上与 Java 小应用程序的竞争结果。ActiveX 控件和 Java 小应用程序在 Web 中的应用在功能方法上非常相似，都是在浏览器中提供可执行的内容。ActiveX 控件与 Java 小应用程序最大的不同就是 ActiveX 控件只能在 Microsoft 的 Windows 操作系统中使用。ActiveX 控件虽然有操作系统平台的限制，但它并不仅仅应用在 Web 应用程序的开发中。ActiveX 控件还可以应用于 Windows 程序设计语言中，如在 Visual Basic 或 Delphi 中都可以使用 ActiveX 控件。

插件是一种与其它种类稍微有点不同的技术，但它仍是浏览器在客户端的扩展。插件实际上是对 Netscape Navigator 浏览器的常规功能进行了扩展，对附加数据类型和其它的一些特点进行支持。一个突出的例子就是我们可以使用插件在 Netscape Navigator 浏览器中显示特定的 MIME（Multipurpose Internet Mail Extension protocol, 多用途的网际邮件扩充协议）类型的文件。在 Netscape 安装文件夹下，Netscape 建立了一个目录 Netscape\Communicator\Pro- gram\Plugins，用于存放在系统中注册了的插件。当我们在浏览器中浏览的内容需要插件来显示时，首先在插件文件夹中搜索是否已经有注册了的插件，如果没有，Navigator 会自动从网络中下载需要的插件，然后进行显示。插件安装或从网络中下载后，就成为了 Navigator 的浏览器的一部分。插件的运行不需要用户的干涉，用户也看不到插件是怎样运行的。插件最主要的作用是对多媒体数据的支持，如播放声音、视频，显示图像等。我们将在后面的章节中学习到如何使用 JavaScript 与插件进行交互作用。

4. 客户端脚本语言

客户端脚本语言也是 Web 客户端的一种重要的技术。JavaScript 现在仍然是最常用的脚本语言，但 Microsoft 也推出了 VBScript。由于使用 Visual Basic 进行编程的开发人员非常多，VBScript 又是 Visual Basic 的一个子集，这些人很容易在不需要进行重新学习的情况下使用 VBScript 来进行 Web 应用程序的开发。我们将在后面的章节中对 JavaScript 和 VBScript 进行比较。

1.4.2 服务器端 Web 技术

服务器端 Web 应用程序结构主要由 Web 服务器和服务器软件扩展组成。服务器的扩展技术很多，图 1-4 显示了服务器端 Web 应用程序结构中各种 Web 技术的关系。

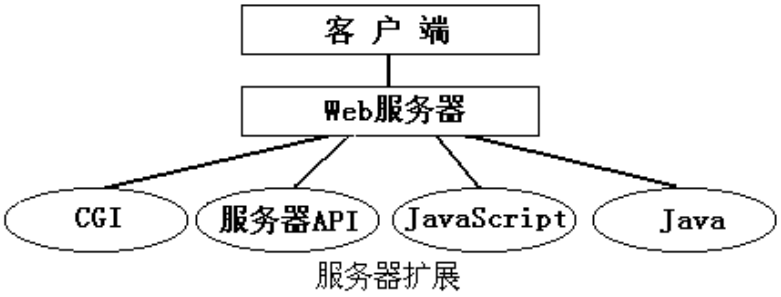


图1-4 服务器端 Web 应用程序结构

1. Web 服务器

Web 服务器实际上是基于 TCP/IP 的应用程序，它的主要作用就是处理客户端的 HTML 文档中的表单发出的请求，然后将结果返回给客户端的浏览器供用户查看。现在主要的几种 Web 服务器有：Netscape Enterprise Server（NES）、Microsoft Internet Information Server（IIS）和 Apche。

2. 服务器扩展

从 Web 服务器自身的角度讲，它只能在浏览器进行请求的情况下向浏览器提供静态的 HTML 文档。为了产生动态的 Web 页，需要对 Web 服务器的功能进行扩展。这些扩展技术包括：CGI、服务器 API、JavaScript 和 Java。

- CGI

CGI 是公用网关接口（Common Gateway Interface）的简称，是在 Web 服务器中使用外部程序的接口方法。我们可以在服务器上运行 CGI 程序或者脚本来产生动态的 Web 内容供客户端用户查看。通常情况下，CGI 程序用于在服务器端处理客户端浏览器中的 HTML 文档中的表单请求。CGI 程序或者脚本被放在 Web 服务器中一个特殊的目录之中，它们接收到客户端的请求，进行处理，然后将结果以 HTML 文档的方式返回到客户端。

CGI 程序的典型应用是用来处理用户发出的数据库查询请求，将查询结果以 Web 页的方式返回给客户端的用户。

在服务器端可用于编写 CGI 程序的程序设计语言很多。只要是能在服务器端运行的程序设计语言（不一定要 Web 服务器的支持）——都可以用来编写 CGI 程序，如 Java、C、C ++、Visual Basic 或 PHP。在 Unix 操作系统中经常使用 Perl 来编写 CGI 程序。

- 服务器 API

服务器 API 是 Web 服务器为编程人员提供的应用程序编程接口（API，Application

Programs Interface）。两种最常用的服务器 API 是 Netscape Server API（NSAPI）和 Microsoft Internet Server API（ISAPI）。API 与服务器是紧密集成的，如在 Windows 环境中可以使用 API 来创建由服务器访问的 DLL（动态链接库），而不是单独的 EXE 文件。

使用服务器 API 的优点是它具有比 CGI 程序更好的性能，因为在使用 CGI 的情况下，当遇到客户端的请求时，每一个请求都会单独运行一遍 CGI 程序，造成资源的浪费，同时也会使数据共享的性能下降。

使用服务器 API 的一个不足的地方是，服务器的 API 是针对某个特定服务器的。不同的服务器具有不同的 API，不同的服务器 API 是很难相互兼容的。比如，使用 ISAPI（IIS 服务器的）编写的 DDL 就不能在 Netscape 服务器中使用。这就限制了使用服务器 API 程序只能在相同的服务器上使用。当然，如果在使用一种服务器的情况下，用不着考虑这种情况了。

- 服务器端的 JavaScript

很多人都知道 JavaScript 可以用来编写客户端的脚本程序，其实 JavaScript 同样可以用在服务器中。在服务器端对 JavaScript 提供支持只有 Netscape 服务器。Netscape 服务器端的 JavaScript 称为 SSJS，可以用来编写由 Netscape 服务器控制的应用程序。JavaScript 在服务器端产生动态 HTML 文档的功能得到了扩展，包括：与客户端通信、在服务器端访问外部文件和连接 SQL 数据库。

Microsoft 也允许用户在 ASP（Active Server Pages）中使用 Jscript 编写的脚本程序。

- Java

最近 Java 在服务器端的应用得到了巨大发展。Sun 公司发布了新的 Java 技术 JavaServer Pages（JSP），从其名称可以看出，JSP 是与 ASP 和 SSJS 进行竞争的技术。现在 Sun 与 Netscape 已结成同盟，称为 iPlanet，准备将 JSP 和 SSJS 融合到一起，构成一个新的、具有超强功能的技术。

另外有一些读者也许听说过 Java Servlet。Servlet 是一种小的组件，有点像在 ASP 中使用的 ActiveX 控件。Servlet 可以在 HTML 文档中被调用或者是用于创建 HTML 文档。Servlet 是为了让用户使用 Java 语言来建立快速、可靠、与平台无关的组件以便创建 Web 应用程序。

1.5 JavaScript 与 VBScript

正如我们在前面提到的一样，JavaScript 并不是惟一可用的 Web 脚本语言。VBScript 和 JavaScript/JScript 一样可以用于编写 Web 脚本应用程序。VBScript、JavaScript 和 JScript 虽然都是处于竞争的地位，但实际上它们之间又有互相补充的作用。在本节我们将对 VBScript 和 JavaScript 进行比较，这将有助于我们学习 JavaScript。

1.5.1 关 于 VBScript

VBScript 是 Microsoft 开发的和 JavaScript 进行竞争的一种 Web 脚本语言。从语言发

展的角度讲，VBScript 和 JavaScript 是完全不同的。我们知道 JavaScript 虽然是从 Java 和 C++发展而来的，但毕竟和 Java 与 C++有许多不同之处，关系也不是非常的紧密。VBScript 和 VBA（Visual Basic for Application）都是 Visual Basic 程序设计语言家族的一员。Visual Basic 是在 Windows 环境中最常用的一种程序设计语言。VBA 主要用于在 Microsoft Office 中编写宏程序或者其它的应用程序。如果你使用 Visual Basic 开发过应用程序，那么使用 VBScript 编写脚本将是非常容易的事。如果你想查看最新的关于 VBScript 的信息，可以访问 Microsoft 的站点：http://msdn.microsoft.com/scripting。

1.5.2 VBScript 的特点

我们可以从下面的几个方面来了解 VBScript 的一些特点。

1. VBScript 是嵌入到 HTML 中的

这一点和 JavaScript 完全相同，VBScript 也是将源代码嵌入到 HTML 文档中的脚本编写语言。VBScript 和 JavaScript 都使用 HTML 标记<script>将脚本程序嵌入到 HTML 文档中。VBScript 使用“text/vbscript”作为<script>的类型属性值。下面是一个简单的使用了 VBScript 的 Web 页。

```
<HTML>
<HEAD>
    <SCRIPT type="text/vbscript">
      alert("这是一个简单的脚本程序！")
    </SCRIPT>
</HEAD>
</HTML>
```

上面的例子是在浏览器中打开该 Web 页时显示的一个消息框（alert）。消息框在 VBScript 和 JavaScript 中都是相同的。将上面的脚本修改成下面的结果。

```
<HTML>
<HEAD>
     <SCRIPT type="text/javascript">
   alert("这是一个简单的脚本程序！")
     </SCRIPT>
</HEAD>
</HTML>
```

2. 对 象 模 型

VBScript 和 JavaScript 一个最重要的相同点是使用的对象模型的层次结构是相同的，只是在不同的版本中有小的差别。对 Web 应用程序开发人员来讲，在不同的情况下，可能需要使用这两种不同的语言。VBScript 和 JavaScript 只是在语法上有一些差别。虽然

VBScript 和 JavaScript 的编程方法不同，但如果要将一个 JavaScript 脚本转换为 VBScript，我们可以直接一句一句地进行转换，而不需要在程序结构上做任何调整。

它们不仅仅在对象模型上相同，VBScript 和 JavaScript 在 HTML 对象的使用上也是相同的。我们知道 JavaScript 脚本是通过 HTML 对象的事件来驱动执行的，VBScript 也是如此。

下面先是一个 JavaScript 的例子，该例的脚本将用户在文本区中输入的字符串转换为大写：

程序清单 1.2　JavaScript 例程

```
<HTML>
<HEAD>
<TITLE>JAVA 实例</TITLE>
<SCRIPT type="text/javascript">
<!--
 function changetext(){
    document.myform.mytext.value=document.myform.mytext.value.toUpperCase()
 }
//-->
</SCRIPT>
</HEAD>
<BODY>
<center>
  <form name="myform">
       <input type="text" name="mytext">
       <br>
       <input type="button" value="转换大写" onclick="changetext()">
  </form>
</center>
</BODY>
</HTML>
```

如果使用 VBScript 来完成相同的功能，则可使用程序清单 1.3 所示程序。

程序清单 1.3　VBScript 例程

```
<HTML>
<HEAD>
<TITLE>script 实例</TITLE>
<SCRIPT type="text/vbscript">
<!--
     sub changetext()
          document.myform.mytext.value=UCase(document.myform.mytext.value)
     end sub
//-->
```

```
</SCRIPT>
</HEAD>
<BODY>
<center>
  <form name="myform">
      <input type="text" name="mytext">
      <br>
      <input type="button" value="转换大写" onclick="changetext()">
  </form>
</center>
</BODY>
</HTML>
```

从两个程序清单可以看出 VBScript 和 JavaScript 在事件驱动执行和对象模型上是完全相同的。两个 Web 页的实际效果如图 1-5 所示。

图1-5 VBScript 和 JavaScript 脚本在运行结果上完全相同

3. 复杂的数据类型

VBScript 和 JavaScript 一样属于弱类型的语言。从表面上看 VBScript 只有一种数据类型(variant),但实际上 VBScript 比 JavaScript 具有更强的数据处理能力。我们可以将 Variant 称之为变体数据类型，该类型的变量可以用于处理各种不同类型的数据。下面例子中的 myvar 是一个 variant 类型的变量，我们先为 myvar 赋一个字符串。

```
myvar="VBScript"
```

在某些情况下，我们可以为 myvar 再赋一个不同类型的值，如：

```
myvar=123
```

这样 myvar 的值从最先的字符串类型变为数值类型。

字符串类型和数值类型实际上是作为 variant 类型的子类型在使用，在表 1-2 中列出了 variant 的所有子数据类型。

表1-2 variant 的子数据类型

子类型	描述
String	字符串类型，最大长度 20 亿个字符
Byte	字节类型，有效值为 0~255
Integer	整数类型，有效值为-32,768~+32,767
Long	长整型，有效值为-2,147,483,648~+2,147,483,647
Single	单精度，有效值为 负数：-3.402823E38~-1.401289E-45 正数：1.401289E-45~3.402823E38
Double	双精度，有效值为 负数：-1.79769313486232E308~-4.94065645841247E-324 正数：4.94065645841247E-324~1.79769313486232E308
Data	日期类型，有效值为 1/1/100~12/31/9999
Boolean	布尔类型，有效值为 ture 或 false

子类型	描述
Empty	空数据，表示未初始化的变量。数值型变量空值为 0，字符串变量空值为空字符串“”
Null	表示没有有效数据的变量，与 Empty 不同
Object	对象，表示 ActiveX 对象
Error	错误，表示 VBScript 的错误编号

4. 不同的过程类型

在 VBScript 中有两种不同的过程类型：子程序和函数。子程序使用 Sub……End Sub 进行定义，子程序和函数最大的特点是没有返回值。函数使用 Function……End Function 进行定义，函数通过函数名返回一个值。

下面的脚本定义了一个子程序 converttext（），converttext（）被调用时，会将文本框中显示的内容修改为“学习使用 VBScript！”

```
<SCRIPT type="text/vbscript">
<!--
    sub converttext()
        document.myform.mytext.value="学习使用 VBScript！"
    end sub
//-->
</SCRIPT>
```

下面是 VBScript 的函数的例子。例子先定义了一个函数 gettext()，该函数通过函数名返回用户在浏览器文本框中输入的内容，然后在 show()子程序中显示函数的返回值。

```
<SCRIPT type="text/vbscript">
<!--
    function gettext()
        gettext=document.myform.mytext.value
    end function

    sub showtext()
        alert(gettext())
    end sub
//-->
</SCRIPT>
```

和 VBScript 不同，JavaScript 中只使用函数。实际上任何一种子程序可以完成的功能，完全可以使用子程序来实现。

1.5.3 一个 VBScript 脚本

为了让你进一步了解如何使用 VBScript 编写脚本程序，下面介绍一个简单的 Web 页。在该页中通过使用 VBScript 脚本显示用户在文本框中输入的两个数的乘积。

程序清单 1.4　使用 VBScript 编写脚本

```
<HTML>
<HEAD>
<TITLE>VBScript 实例</TITLE>
<SCRIPT type="text/vbscript">
<!--
    sub getMultiply()
        num1=document.myform.mytext1.value
        num2=document.myform.mytext2.value
        result=num1*num2
        alert(result)
    end sub
//-->
</SCRIPT>
</HEAD>
<BODY>
<center>
  <h3>使用 VBScript 在客户端完成数据处理</h3>
  <br>
  <form name="myform">
      第一个数：<input type="text" name="mytext1">
      <br><br>
      第二个数：<input type="text" name="mytext2">
      <br><br>
    <input type="button" value="显示两个数的积" onclick="getMultiply()">
  </form>
</center>
</BODY>
</HTML>
```

为了和 VBScript 进行比较，程序清单 1.5 列出的脚本可以完成和程序清单 1.4 中脚本相同的功能。

程序清单 1.5　使用 JavaScript 编写脚本

```
<HTML>
<HEAD>
<TITLE>JavaScript 实例</TITLE>
<SCRIPT type="text/javascript">
<!--
    function getMultiply(){
        num1=document.myform.mytext1.value
        num2=document.myform.mytext2.value
        result=num1*num2
```

```
            alert(result)
        }
//-->
</SCRIPT>
</HEAD>
<BODY>
<center>

  <h3>使用 JavaScript 在客户端完成数据处理</h3>
  <br>
  <form name="myform">
        第一个数：<input type="text" name="mytext1">
        <br><br>
        第二个数：<input type="text" name="mytext2">
        <br><br>
     <input type="button" value="显示两个数的积" onclick="getMultiply()">
  </form>
</center>
</BODY>
</HTML>
```

第2章 JavaScript 与 HTML

主要内容

- HTML 基　础
- 将 JavaScript 脚本嵌入 HTML 文档
- 编写自己的 JavaScript 脚本

本章导读

HTML 在众多的 Web 技术中恐怕是最显得微不足道的了，但 HTML 的的确确是精彩的 Web 世界必不可少的基石。使用 HTML 可以创建静态的、内容丰富的、有趣的 Web 页，这在 Internet 世界的初期可说是一个创举。使用 JavaScript 可以为 HTML 文档增加交互性、增强对用户操作的反应能力、使 Web 页更加具有吸引力。

换一个角度，可以将 JavaScript 看做是 CGI 程序的一个发展的新阶段。CGI 是基于服务器端的应用程序，而 JavaScript 则是基于客户端的。通过使用 JavaScript，Web 开发人员可以编写脚本，然后将脚本嵌入到 HTML 文档中来创建动态的 Web 页。JavaScript 将原来需要使用 CGI 程序在服务器端完成的处理操作放在客户端来完成，这样大大减少对用户操作的响应时间，使用户可以在更短的时间内完成更多的事。

我们知道 JavaScript 是嵌入到 HTML 文档中的，JavaScript 代码的运行在很多时候也需要通过 HTML 对象的事件进行驱动。所以无论对于 Web 开发人员还是阅读 HTML 文档，了解 HTML 的基本知识还是很重要的。

2.6 HTML

开发简单的 HTML 文档、标签和表单是件非常容易的事。在 HTML 中的大多数标签都有一些复杂的属性，使用复杂的标签和属性可以创建更复杂、可用性更强的 Web 页。

HTML 标准始终在不断地变化，当前的 HTML 版本为 4.01。HTML 的一个新的版本名称将是 XHTML，不过这还需要得到 Word Wide Web（W3C）的同意。Web 开发人员可以随时从 W3C 的 Web 站点 http://www.w3.org 中查看有关 HTML 的各种信息。如图 2-1 所示。

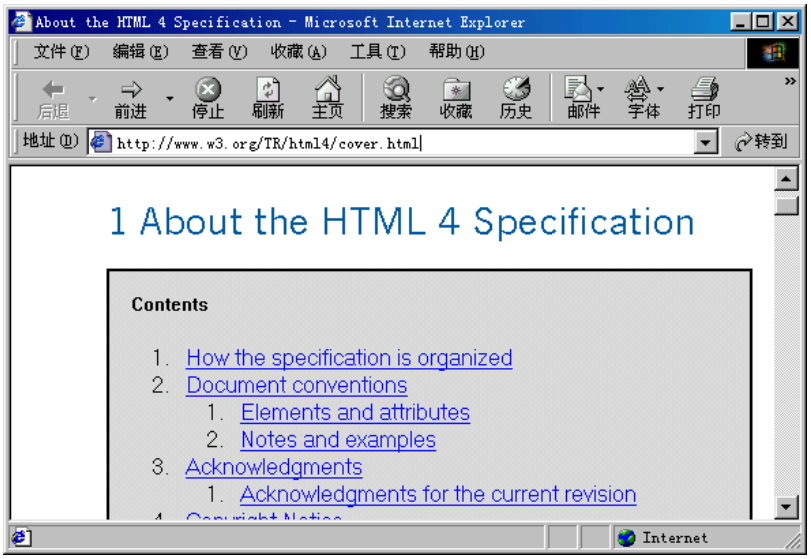


图2-6 W3C 上的 HTML 信息

HTML 的标准和浏览器都是在不断发展，当我们在使用 HTML 标签，特别是一些新的标签时，必须要确保浏览器能够支持它。另外，在不同的浏览器中，也会出现一些非标准的标签。如<marquee>标签只能在 Internet Explorer 中使用，Navigator 是不支持的。而 Navigator 中的<layer>和<ilayer>同样也不能在 Internet Explorer 中使用。

所以，当我们在编写 HTML 文档时，应充分考虑多浏览器的因素，能让多种不同浏览器都能阅读是衡量 HTML 文档好坏的一个标准。

2.6.2 HTML 基本知识

在客户端的 JavaScript 脚本和 HTML 文档的关系是非常密切的，所以在我们编写脚本之前，必须先熟悉 HTML 文档的结构和 HTML 标签。当对 HTML 掌握后，编写 Web 页将是非常简单的事。通常，一个 Web 页是由文本、标签和一些注释组成的。

Web 页中的文本是最简单、最容易理解的。输入到 Web 页中的文本是我们需要向用户展示的内容。标签则是 Web 页中最重要、需要花时间学习的东西。一个 HTML 标签通常是由“<”开头，由“>”结尾的标志。标签的作用就是告诉浏览器该如何显示我们书写的文本。

HTML 标签有单独标签和成对标签。单独标签如<hr>和
，它们都是单独使用的。<hr>的作用是显示一条水平直线，
是用于进行换行。单独标签通常又称之为空标签，空标签只能单独使用，不能用于格式化文本。

成对标签又称之为容器，它使用一个开始标签和一个结束标签来标识文本。结束标签是在一个标签的名称前加一个“/”。如下面使用的“<title>”标签：

<title>学习 JavaScript</title>

在<title>和</title>之间的是容器标识的文本，“学习 JavaScript”被标记为一个 HTML 文档的标题。

一些复杂的标签还可以通过设置属性来控制标签的显示效果。如：<hr width=200 color=red>定义了一条长 200 个像素，颜色为红色的水平线。很多 HTML 标签都具有自己的属性。

还有一些特殊的标签，它们必须放在其它的容器之中才能够使用。比如<input>必须放在<form>标签内；<li>必须放在<ol>标签内。所有的其它 HTML 标签都放在<html>和</html>之间。

在 HTML 文档中，如果需要，有时会加入适当的注释对文档进行说明。注释的内容不会显示在浏览器中。在 HTML 文档中加入注释使用“<!--注释-->”，例如：

<!--使用 JavaScript 方法-->

2.6.3 HTML 文档结构

HTML 文档的结构是非常重要的，标签在文档中放置的位置的不同，会有不同的显示效果。最简单的 HTML 文档结构如下：

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>
</BODY>
</HTML>
```

在<HTML>和</HTML>之间放置了 HTML 文档的所有内容。在<BODY>和</BODY>

之间放置的是实际要显示的文本内容和其它用于控制文本显示方式的标签。

下面是一个简单的 HTML 文档：

```
<html>
  <head>
    <title>一个简单的 HTML 文档</title>
    <script type="text/JavaScript"></script>
  </head>
<body>
  <h1>简单的 HTML 文档</h1>
  <ol type=1>
      <li>Java</li>
      <li>JavaScript</li>
      <li>Visual Basic</li>
      <li>VBScript</li>
   </ol>
   <table width="3" height="3" border>
      <tr>
        <th colspan="12">表格</th>
      </tr>
      <tr>
        <td>One</td>
        <td>Two</td>
        <td>Three</td>
      </tr>
      <tr>
        <td>Four</td>
        <td>Five</td>
        <td>Six</td>
      </tr>
   </table>
<p>
   <form>
      <input name="button1" type="submit">
   </form>
</p>
这是一个简单的 HTML 文件!
</body>
</html>
```

该 HTML 文档在 Microsoft Internet Explorer 中的实际效果如图 2-2 所示。

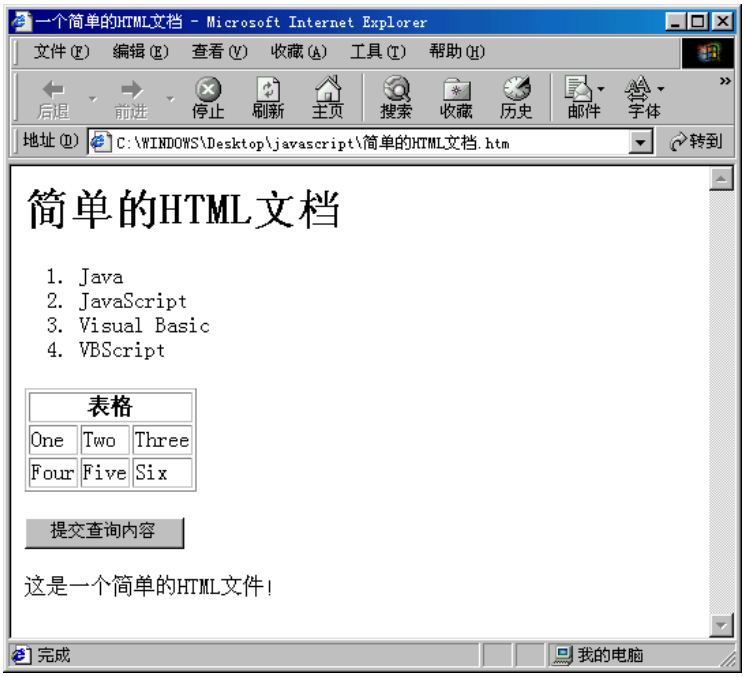


图2-7 一个简单的 HTML 文档

2.6.4 公 共 属 性

在前面我们已经提到 HTML 的许多标签都具有属性，属性用于控制标签怎样操作和如何显示。并不是所有的标签都具有属性，有一些标签具有公共的属性，这些属性的名称和作用都完全相同。在表 2-1 中列出了 HTML 标签的共同属性和作用。

表2-3

属性	作用
class	指明 HTML 标签所属的类（class）
id	定义一个惟一的标识，在整个文档中标识该标签
name	为标签命名，该名称可以在 JavaScript 或 CGI 脚本中引用
style	指明标签使用的样式（style）

2.7 在 HTML 文档中嵌入 JavaScript

JavaScript 代码使用<script>和</script>集成到 HTML 文档中。在一个 HTML 文档中，可以有多对<script>和</script>来嵌入多段 JavaScript 代码，每个 JavaScript 代码中可以包含一条或多条 JavaScript 语句。

<script>也有几个属性，它们的作用如表 2-2 所示。

表2-4

属性	作用
defer	属性值为 boolean 值，用于告诉浏览器脚本是否有输出内容
language	用于指定脚本使用的语言，现在已经很少使用该属性
src	指定使用的外部 JavaScript 代码的 URL 地址
type	代替 language 属性，用于指明脚本使用的语言

- Defer

Defer 是一个简单的属性，它的作用就是告诉浏览器 JavaScript 代码是否会产生输出，也可以说 document.write()方法是否被使用。下面是一个简单的例子：

```
<script type="text/JavaScript" defer>
<!--
    //仅仅声明了一个变量，没有输出语句
    var myInt=2000
//-->
</script>
```

- language

language 属性在最近的 HTML 和 XHTML 中已经不再使用，但为了使以前的 Web 也仍能够不加修改就可以在浏览器中正确地显示，所以现在的<script>仍保留了 language 属性。

language 属性的使用格式如下：

<script language ="JavaScript">

该标签告诉浏览器脚本使用的是 JavaScript 1.0。如果是使用其它版本的 JavaScript，如 1.1，可以按照如下格式使用：

<script language ="JavaScript 1.1">

language 属性的值告诉浏览器应该按照哪一种语言的版本来执行脚本代码。在表 2-3 中列出了 language 属性值可用的 JavaScript 版本和支持的浏览器。

表2-5

language 属性值	支持的浏览器
JavaScript	Navigator 2+和 Internet Explorer 3+
JavaScript 1.1	Navigator 3+和 Internet Explorer 3+
JavaScript 1.2	Navigator 4+和 Internet Explorer 4+
JavaScript 1.3	Navigator 4.05+和 Internet Explorer 5+
JavaScript 1.4	HotJava 3+和早期 alpha 版本的 Mozilla/Navigator 5（pre_M12）
JavaScript 1.5	Mozilla/Navigator beta 1.0（M12+）

如果在<script>中没有使用 language 属性，则浏览器会假设使用的是 JavaScript 1.0。我们可以利用 language 属性，在 Web 页中编写多种版本的程序代码，这样浏览器可以通过判别语言的版本来执行它支持的代码，同时会忽略它不支持的代码。

下面的 HTML 文档可以用于测试浏览器支持的 JavaScript 版本。

```
<html>
<head>
  <title>JavaScript 版本测试</title>
```

```
</head>
<body>
<script language="JavaScript">
  //仅支持 JavaScript 1.0 的浏览器才读该部分
  document.write('浏览器支持 JavaScript 1.0<br>');
</script>
<script language="JavaScript1.1">
  //仅支持 JavaScript 1.1 的浏览器才读该部分
  document.write('浏览器支持 JavaScript 1.1<br>');
</script>
<script language="JavaScript1.2">
  //仅支持 JavaScript 1.2 的浏览器才读该部分
  document.write('浏览器支持 JavaScript 1.2<br>');
</script>
<script language="JavaScript1.3">
  //仅支持 JavaScript 1.3 的浏览器才读该部分
  document.write('浏览器支持 JavaScript 1.3<br>');
</script>
<script language="JavaScript1.4">
  //仅支持 JavaScript 1.4 的浏览器才读该部分
  document.write('浏览器支持 JavaScript 1.4<br>');
</script>
<script language="JavaScript1.5">
  //仅支持 JavaScript 1.5 的浏览器才读该部分
  document.write('浏览器支持 JavaScript 1.5<br>');
</script>
</body>
</html>
```

- Src

将 JavaScript 代码嵌入 HTML 文档有两种方式：一是直接将代码书写在 HTML 文档中；另一种方式是使用<script>的 src 属性。第一种方式我们称之为内联方式。使用<script>的 src 属性是在一个 HTML 文档中引用另一个文件中的 JavaScript 代码——将 JavaScript 代码存放于一个单独的.js 文件中。

在下面的 HTML 文档中引用了在另一个文件 testscript.js 中的 JavaScript 代码。

```
<HTML>
<HEAD>
<TITLE>JAVA 实例</TITLE>
<SCRIPT src="testscript.js">
</SCRIPT>
</HEAD>
```

<BODY>

</BODY>

</HTML>

在该 HTML 文档中没有任何将在浏览器中显示的信息，但通过引用另一个文件中的 JavaScript 代码，在浏览器窗口中输出了一段信息。被引用的 testscript.js 文件内容如下：

document.write（“这里引用了其它文件中的 JavaScript 代码”）

使用 src 属性的一个好处是可以在不打开 HTML 文件的情况下修改 JavaScript 代码，但它的一个缺陷是代码的修改可能影响到 HTML 文档，比如当修改了.js 文件中的函数名之后，就必须修改 HTML 文档中用到的函数名，否则 HTML 文档将会出错。

使用 src 属性的另一个好处是可以保护您不想让别人看到的 JavaScript 代码。

- type

language 是最初的属性，它不仅用于说明使用的语言，如 JavaScript 或 VBScript，同时也可以说明语言的版本，如 JavaScript 1.1 或 JavaScript 1.5。在最近的 HTML 和 XHTML 中已不再使用 Language 属性，取而代之的是 type 属性。

type 属性的使用格式如下：

<script type="text/JavaScript">

或

<script type="text/vbscript">

HTML 4.01 标准建议用户在<head>内放置一个<meta>标签来说明 HTML 文档中的所有脚本语言使用的默认语言类型，格式如下：

<meta http-equiv="Content-Script-Type" content="text/JavaScript">

这里可以使用"text/vbscript"来代替"text/JavaScript"。如果使用了 Type 属性，则 Type 属性将优先于<meta>中设置的默认语言类型。

2.8 编写 JavaScript 脚本

2.8.1 编写脚本需要什么工具

实际上编写 JavaScript 脚本是一件简单的工作，花心思的是我们怎样才能将我们的 Web 页做得更有趣、更具有吸引力。编写脚本对工具的选择没有什么特别的要求，不过我们还是需要加以选择。为了编写 Web 页，我们至少需要三种工具：

- HTML 文档编写工具;
- JavaScript 脚本编写工具;
- 查看 Web 页和脚本效果的浏览器。

无论是 HTML 文档编写工具、JavaScript 脚本编写工具还是浏览器都是多种多样的。从学习 JavaScript 的角度讲，我认为使用 Windows 中的记事本和 Internet Explorer 浏览器就可以了，这两样工具就在 Windows 中，拿来使用就行了。另外，学习本书中的某一些

例子，您可能还需要有 Netscape Navigator。

2.8.2 编写您的第一个脚本

您现在要做的是使用 JavaScript 脚本在浏览器中显示一行文本“看！这是我的第一个脚本！”。要显示这一行文本的脚本很简单，可以使用下面的 JavaScript 语句：

document.write("看！这是我的第一个脚本！")

要将这一句 JavaScript 脚本嵌入到 HTML 中，最后我们才能看到其效果，可以按照如下步骤进行：

（1）编写 HTML 文档。

在 Windows 中打开记事本，在记事本中输入如下内容：

```
<html>
<head>
<title></title>
</head>
<body>
</body>
</html>
```

这是最简单的 HTML 文档，它在浏览器中不会显示任何结果。

（2）插入<SCRIPT>……</SCRIPT>标签。

<SCRIPT>……</SCRIPT>标签从理论上讲，可以插入到 HTML 文档中的任何位置，比如在<head>和</head>之间或是<body>和</body>之间。按许多人的习惯，常将<SCRIPT>……</SCRIPT>标签放在<head>和</head>之间，如下所示：

```
<html>
<head>
<title></title>
<SCRIPT TYPE="text/JavaScript">
    <!--
     //在此插入 JavaScript 脚本
     //-->
</SCRIPT>
</head>
<body>
</body>
</html>
```

这是插入了<SCRIPT>……</SCRIPT>标签的 HTML 文档，这个文档具有一定的使用价值。您可以将它作为模板保存起来，在以后的学习中使用。

（3）书写脚本。

加入脚本之后的 HTML 文档如下所示：

```
<html>
```

```
<head>
<title></title>
<SCRIPT TYPE="text/JavaScript">
    <!--
     document.write("看！这是我的第一个脚本！")
     //-->
</SCRIPT>
</head>
<body>
</body>
</html>
```

（4）保存 HTML 文档。

因为记事本保存文件的默认类型为.txt 文件，所以在保存编写的 HTML 文档时，应先将文件类型选择为“所有文件”，再将文件保存为.htm 或.html 文件。

（5）浏览效果。

将编写的文件保存好后，在浏览器中打开该文件，查看实际效果。在前面编写的 HTML 在 Internet Explorer 中的实际效果如图 2-3 所示。

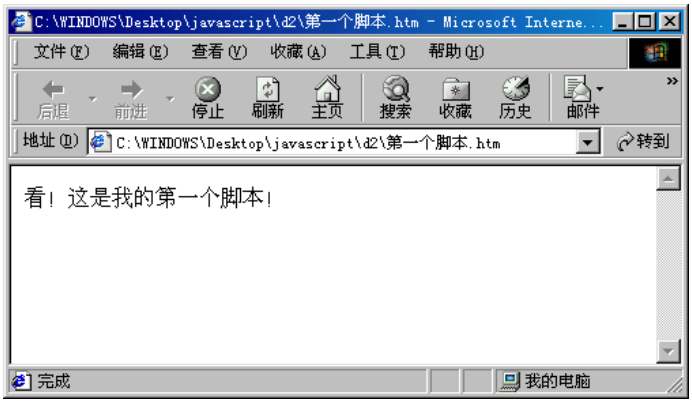


图2-8 第一个脚本的效果

2.8.3 脚本在什么时候执行

根据 JavaScript 脚本编写的方式，脚本的执行有多种情况。当浏览器打开一个 HTML 文档时，它将从头开始解释整个文档，像在上例中使用 document.write（）方法在遇到它的时候执行；而有一些脚本，如函数（function），则会在它们被调用的时候运行。脚本函数的调用往往都是通过事件来进行驱动的，如在一个 HTML 文档被装载（onLoad）的时候可以执行脚本函数。

1. 在打开页面时执行脚本

当浏览器打开一个 HTML 文档时，它会从头开始解释整个文档，包括 HTML 标签和脚本。如果脚本中有可以直接执行的语句，则会在遇到的时候马上解释执行。下面的脚本中的 alert（）语句会在页面打开时立即执行，显示一个消息框，在用户关闭了对话框之后才会显示其它的页面内容。

```
<html>
<head>
<title>立即执行</title>
<script type="text/JavaScript">
```

```
    <!--
      alert("立即执行了脚本");  //首先会显示的消息内容
      // -->
</script>
</head>
<body>
  <b>浏览器在解释 HTML 文档时执行了 JavaScript 脚本
  </b>
</body>
</html>
```

该 HTML 文档在浏览器中打开时的实际效果如图 2-4 所示。

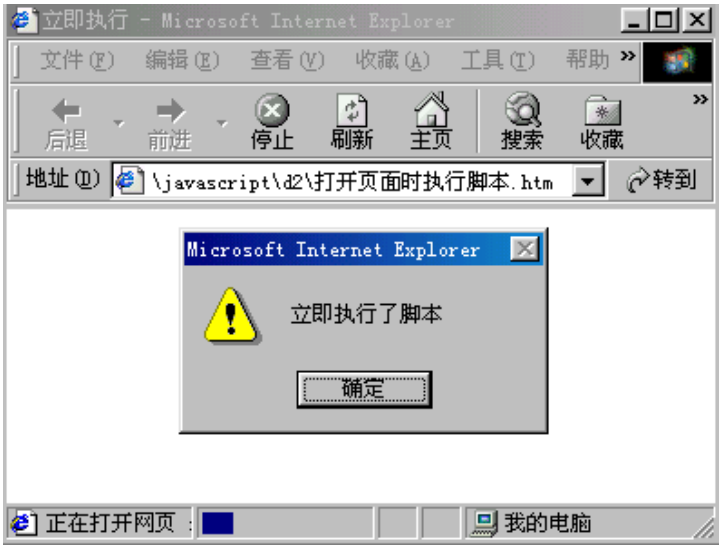


图2-9 浏览器在显示页面内容前执行了脚本

2. 利用 onLoad 事件执行脚本

onLoad 事件是一个页面在浏览器中打开时发生的，该方法常用于在打开一个页面的同时向用户显示一些消息。在下面的例子中，脚本定义了一个 opennews（）函数，该函数如果不被调用，它将不会被执行。我们可在<body>标签的 onLoad 事件中调用该函数，如下所示：

```
<html>
<head>
<title>利用 onload 执行脚本</title>
<script type="text/JavaScript">
  <!--
    function opennews(){
      alert("利用 onLoad 事件执行脚本")
```

```
    }
   // -->
</script>
</head>
<body onload="opennews()">
  <b>利用 onLoad 事件执行脚本
    </b>
</body>
</html>
```

该 HTML 文档在浏览器中打开时的实际效果如图 2-5 所示。

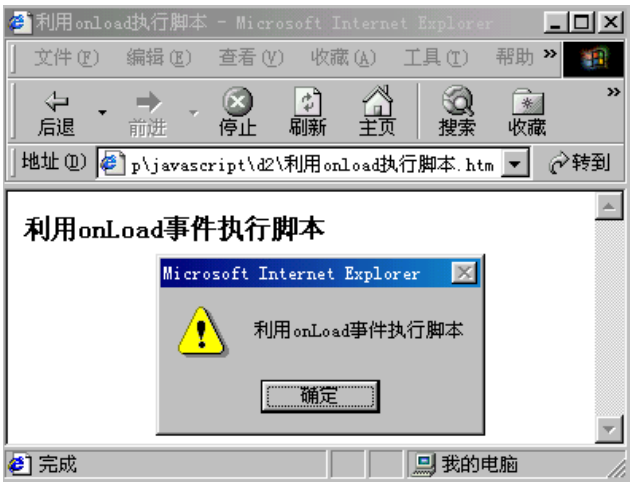


图2-10 浏览器在显示页面内容的同时显示消息框

3. 通过用户事件执行脚本

用户在浏览器中阅读 Web 页时经常会使用鼠标和键盘，比如移动鼠标、点击链接、单击按钮等行为，这些行为都会产生相应的事件。我们可以通过对这些事件编写脚本来进行特殊的处理。

在下面的例子中，在脚本中定义了一个函数 shownews（），函数被调用时显示一个消息框。然后在 HTML 的表单中定义了一个按钮，当单击该按钮时，shownews（）将被调用。

```
<html>
<head>
<title>通过用户事件执行脚本</title>
<script type="text/JavaScript">
    <!--
        function shownews(){
            alert("单击按钮时执行脚本！ ");
```

```
        }
    // -->
</script>
</head>
<body>
  <b>通过用户事件执行脚本</b>
<form method="send">
    <input type="Button" name="BUTTON1" value="单击"
          onclick="shownews()">
</form>
</body>
</html>
```

该 HTML 文档在浏览器中打开时的实际效果如图 2-6 所示。

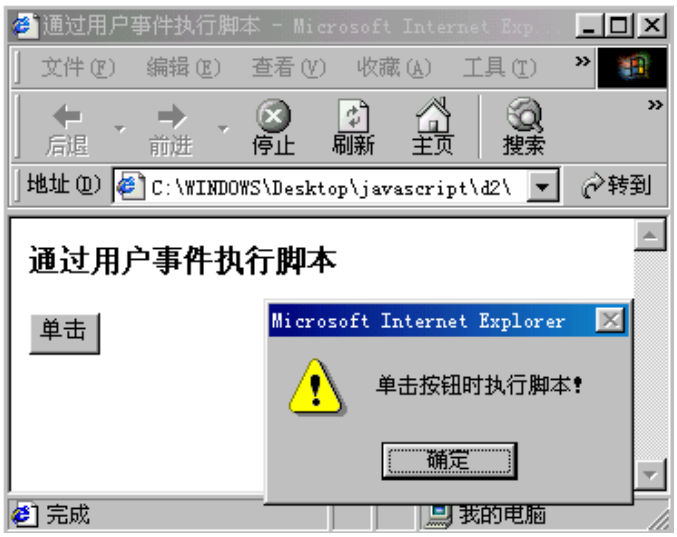


图2-11 单击按钮后才会执行脚本显示消息框

第3章 JavaScript 基本语法

主要内容

- JavaScript 数据类型
- JavaScript 运算符和表达式
- JavaScript 控制结构和循环

本章导读

本章主要介绍 JavaScript 的基本语法。由于 JavaScript 是由 C 语言演变而来的，所以对于学过 Java、C、C++的读者可以跳过本章去继续学习后面的章节。本章除了对 JavaScript 的数据类型、运算符和表达式做了简要的介绍外，着重介绍了条件语句、循环语句等重要内容，并配以详细实例来演绎。

3.9 JavaScript 基本数据结构

JavaScript 语言同其它语言一样，有它自身的基本数据类型、表达式和算术运算符以及程序的基本框架结构。JavaScript 在很大程度上借用了 Java 的语法，而 Java 又是由 C 和 C++演生而来的，所以 JavaScript 和 C 有许多相似的语法特点，并且抛弃了 C 语言中有关指针等容易产生错误的内容。对于已经具备 C++或 C 语言基础或有 C 语言编程经验的人来说，学习 JavaScript 语言是非常轻松容易的。

3.9.1 基本数据类型

在 JavaScript 中有四种基本的数据类型：

- **数字型**：整数和实数;
- **字符串型**：用“”号或 ‘’ 括起来的字符或数值;
- **逻辑型**：用 True 或 False 表示;
- **空值**：表示什么也没有。

在 JavaScript 的基本类型中的数据可以是常量，也可以变量。由于 JavaScript 采用弱类型的形式，因而一个数据的变量或常量不必首先作声明，而是在使用或赋值时确定其数据的类型。当然也可以先声明该数据的类型，它是通过在赋值时自动说明其数据类型的。

3.9.2 常　　量

在程序运行过程中，其值不能被改变的量称为常量。常量可根据不同类型分为整型常量、实型常量、字符型常量等。

1. 整型和实型常量

整型常量由整数表示，如 123、-2000，也可以使用十六进制、八进制表示其值。实型常量是由整数部分加小数部分表示，如 123.45、-3.1415，也可以使用科学或标准方法表示，如 3e6、4E-10。

2. 字符型常量

使用双引号（“ ”）或单引号（‘ ’）括起来的一个字符或字符串，如“JavaScript”、“1234.5”、“sadfsdf_1412412”、‘sdkfsdkf’等。

3. 逻 辑 常 量

逻辑常量只有 True（真）或 False（假）两种值。它往往用做程序的判断条件。

4. 空　　值

JavaScript 还提供一个空值：null，表示什么也没有。

5. 特 殊 字 符

JavaScript 提供一些同样以反斜杠“\”开头的不可显示的特殊字符。通常称为控制字符。例如：‘\n’表示换行，‘\b’表示退格，‘\r’表示回车，‘\f’表示换页等，与 C 语言很相似。

3.9.3 变　　量

在程序运行过程中，其值可以改变的量称为变量。一个变量由变量名和变量值构成。对于变量必须明确变量的命名、变量的声明、变量的类型以及变量的作用域。

1. 变量的命名

JavaScript 中的变量名必须以字母（a~z 或 A~Z）开头，其后可以跟数字和下划线等，但不能有空格、“＋”、“－”、“，”“？”等运算符或标点符号作为变量名。另外，对于 JavaScript 中的关键字不能作为变量名，如 Var、int、double、false、true 等都不能作为变量的名称。


在对变量命名时，应尽量使变量名有意义，从而增加程序可读性，减少错误发生。

2. 变量的声明

一般在使用一个变量前，应该事先对变量进行声明。在 JavaScript 中，变量可以用命令 Var 作声明，例如声明一个名为 number1 的变量，其声明格式如下：

```
<script>
var number1
</script>
```

在变量声明时可以同时给该变量赋初值，例如让变量 number1 的初值为 100，具体如下：

```
<script>
var number1=100
</script>
```

另外，在同一行中可以声明多个变量，变量与变量之间用逗号隔开。例如在同一行中声明变量 number1、number2、number3，具体如下：

```
<script>
```

```
var number1，number2，number3
</script>
```

在 JavaScript 中，变量也可以不作声明，而在使用时再根据数据的类型来确其变量的类型。

3. 变量的类型

JavaScript 是一种弱类型的语言，变量的类型不像其它语言那样规定得比较死，对于同一变量可以赋不同类型的变量值。例如：

```
<script>
 var number1=100
 number1=“number1 as string”
</script>
```

其中，变量 number1 先被赋初值为 100，此时 number1 的类型是整型。接下来又让 number1 等于一个字符串“number1 as string”，此时 number1 的类型又变成了字符型。这些在 JavaScript 语言中都是允许的，所以一般可以不用关心变量是什么类型。

4. 变量的作用域

变量的作用域是指变量在什么范围内才有效。在 JavaScript 中同样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；而局部变量是在函数体内定义的，只在该函数范围内有效，而对其它函数则是无效。

3.10 JavaScript 运算符和表达式

JavaScript 中运算符有很多，例如有算术运算符，如＋、－、*、/等；有比较运算符如!=、＝=等；有逻辑运算符如!、||等。这些运算符的用法与 C 语言很相似。

3.10.1 算术运算符和表达式

通常可以用一组常量、变量、运算符构成一个表达式，通过表达式可以返回一个单一的结果值，这个值可以是数值、字符串、逻辑值等。例如表达式：5+6，其返回值为 11。

1. 基本算术运算符

JavaScript 中基本算术运算符有：

- +：加法运算符，或正值运算符。如 3 + 5、 + 6;
- －：减法运算符，或负值运算符。如 9-3、-7;

- *****：乘法运算符。如 4*6;
- **/**：除法运算符。如 8/4;
- **%**：求模运算符。如 5%2。

2. 递增运算符（＋＋）

递增运算符的作用是把操作数的值加 1 后传回，有两种使用格式，具体如下：

var＋＋ 和 ＋＋var

如果使用 var ＋ ＋格式，则操作数的值将被加 1 传回，而该表达式的返回值不变。例如下面的表达式：

y = x＋＋

如果 x 的初值为 5，则执行后的结果是 x 为 6，而 y 为 5。

如果使用 ＋＋ var 格式，则操作数的值将被加 1 传回，而该表达式的返回值也将加 1。例如下面的表达式：

y = ＋＋x

如果 x 的初值为 5，则执行后的结果是 x 为 6，而 y 为 6。

3. 递减运算符（--）

递减运算符的作用是把操作数的值减 1 后传回，有两种使用格式，具体如下：

var -- 和 -- var

如果使用 var --格式，则操作数的值将被减 1 传回，而该表达式的返回值不变。例如下面的表达式：

y = x--

如果 x 的初值为 5，则执行后的结果是 x 为 4 ，而 y 为 5。

如果使用 --var 格式，则操作数的值将被减 1 传回，而该表达式的返回值也将减 1。例如下面的表达式：

y = --x

如果 x 的初值为 5，则执行后的结果是 x 为 4 ，而 y 为 4 。

3.10.2 比较运算符

比较运算符的作用是比较两边操作数，并将比较结果返回成逻辑值（true 或 False）。具体如下：

- **>**：当左边操作数大于右边操作数时返回 true，否则返回 False;
- **<**：当左边操作数小于右边操作数时返回 true，否则返回 False;
- **>=**：当左边操作数大于等于右边操作数时返回 true，否则返回 False;
- **<=**：当左边操作数小于等于右边操作数时返回 true，否则返回 False;
- **= =**：当左边操作数等于右边操作数时返回 true，否则返回 False;
- **! =**：当左边操作数不等于右边操作数时返回 true，否则返回 False。

例如：表达式 4<5 返回 true，表达式 4>5 返回 False。

3.10.3 逻辑运算符

逻辑运算符采用逻辑值（true 或 False）作为操作数，其返回值也是逻辑值，具体如下：

- **&&**：逻辑与，当左右两边操作数都为 true 时返回值为 true，否则返回 False;
- **||**：逻辑或，当左右两边操作数都为 False 时返回值为 False，否则返回 true;
- ！：逻辑非，当操作数为 true 时返回值为 False，否则返回 true;
- **?**：：三目操作符使用的主要格式如下:
  - 操作数？结果 1：结果 2;
  - 若操作数的结果为真，则表述式的结果为结果 1，否则为结果 2。

3.10.4 按位操作运算符

按位操作运算符包括按位逻辑运算符和按位移动运算符。

- **按位逻辑运算符**：&（按位与）、|（按位或）、∧（按位异或）;
- **按位移动运算符**：<<（左移）、>>（右移）、>>>（右移，零填充）。

3.10.5 赋值运算符

赋值运算符的作用是将一个值赋给一个变量。最常用的赋值运算符是“=”，并由“=”赋值运算符和其它一些运算符复合产生一些新的赋值运算符，如+=、*=等。

- ＋＝：将变量与所赋的值相加后再赋给该变量，例如：a＋＝5 等价于 a＝a＋5;
- －＝：将变量与所赋的值相减后再赋给该变量，例如：a－＝5 等价于 a＝a－5;
- *＝：将变量与所赋的值相乘后再赋给该变量，例如：a＊＝5 等价于 a＝a＊5;
- ／＝：将变量与所赋的值相除后再赋给该变量，例如：a／＝5 等价于 a＝a／5;
- ％＝：将变量与所赋的值求模后再赋给该变量，例如：a％＝5 等价于 a＝a％5。

另外，还有一些与位运算有关的运算符：

- **<<**＝：将变量各位左移若干位后再赋给该变量，例如 a<<＝2 等价于 a＝a<<2，表示将变量 a 的各位左移 2 位，再赋给该变量 a;
- **>>**＝：将变量各位右移若干位后再赋给该变量;
- **&**＝：将变量与所赋的值按位与后再赋给该变量;
- ∧＝：将变量与所赋的值按位异或后再赋给该变量;
- ｜＝：将变量与所赋的值按位或后再赋给该变量。

3.11 JavaScript 控制结构和循环

在任何一种语言中，程序控制流是必须的，它能使得整个程序减小混乱，使之顺利按其一定的方式执行。JavaScript 常用的程序控制流结构及语句包括：条件语句、循环语句以及其它一些语句。

3.11.1 条 件 语 句

1. If 语 句

if 语句是使用最为普遍的条件语句，每一种编程语言都有一种或多种形式的该类语句，并在编程中总是避免不了要用到它。if 语句基本格式如下：

```
if (conditional) {
   [Statements]
}
```

其中 conditional 表示条件，它可以是任何一种逻辑表达式。如果条件的返回结果为 true，则先执行条件后面的 Statements，Statements 表示语句组，然后再执行后面的程序；反之，Java 程序则直接跳过条件后面的 Statements，去执行后面的程序。

下面是一个 if 语句使用的例子：

```
<html>
<head>
<title></title>
</head>
<body>
<script type="text/javascript">
<!--
//declare variables
var visitorInterest
//visitorInterest="新产品"
visitorInterest="技术支持"
document.writeln("您好！欢迎进入服务区")
    if (visitorInterest="新产品"){
        document.writeln("最新产品开始介绍。")
    }
if (visitorInterest="技术支持"){
    document.writeln("技术支持已经开启。")
    document.writeln("在这之前，让我为您介绍我们的最新产品。")
```

```
    }
document.writeln("我们的新产品很安全实用。")
//-->
</script>
</body>
</html>
```

本例的执行结果如图 3-1 所示。

图3-12

在编程过程中，大括号“{}”的使用是非常有用的，它有助于增强程序的逻辑性和条理性，尤其在嵌套使用 if 语句时。下面是一个嵌套使用 if 语句的例子：

```
<html>
<head>
  <title>JavaScript Unleashed</title>
</head>
<body>
  <script type="text/javascript">
<!--

    // 变量声明
var needsInfo;
var needsMoreInfo;
needsInfo = true;
needsMoreInfo = true;

document.writeln("这是本公司的在线服务。");
if (needsInfo) {
   document.writeln("<br>打开技术支持。");
```

```
    if(needsMoreInfo){
      document.write("<br>本公司的最新产品介绍。");
    }
    document.writeln("<br>欢迎使用本公司的在线服务。");
    }
  // -->
  </script>
  </body>
  </html>
```

嵌套使用 if 语句时，程序首先对第一个 if 语句进行条件判断，如果条件为真，则执行下面的语句并对嵌套的 if 语句再进行判断。如果第一个 if 语句条件为假，则直接跳过该条件的所有语句，去执行后面的语句，也就是说，只要 if 语句条件为假，它后面的 if 语句的真假已没有关系。本例的执行结果如图 3-2 所示。您还可以通过重新设置变量 needsInfo 和 needsMoreInfo 的值来得到 3 种不同的结果。

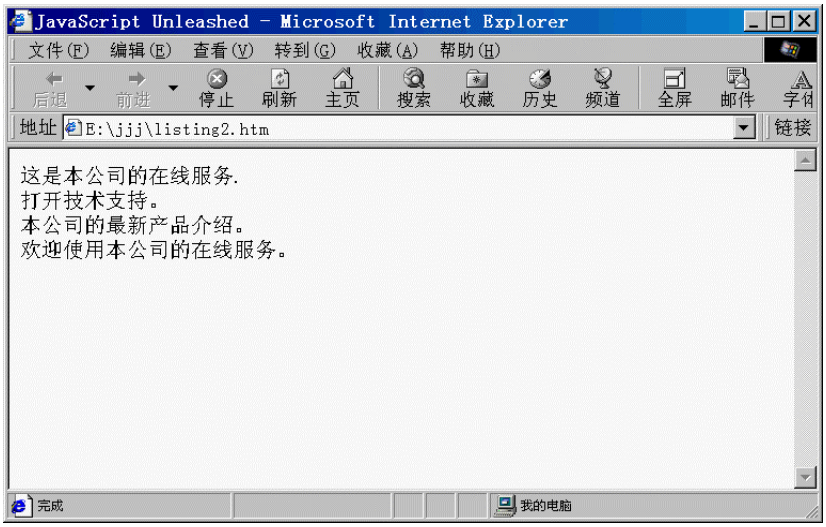


图3-13

2. if..else 语句

有时，仅仅使用 if 语句还不够，在条件表达式的值为假的情况下，您可以保留一系列可执行的语句。您只需按如下所示在 if 语句块后面添加 else 语句块就行了，这种语句的格式如下：

```
if（conditional）{
Statements
} else{
    Statements
}
```

另外，如果您不想直接进入 else 默认块，您还可以用另外的 if 语句来合并 else 部分。用这种方法，您可以在执行适当操作之前估计几种不同的可接受的情况。这种语句的格式

如下：

if（conditional）{
Statements
} else if (conditional) {
Statements
} else {
Statements
}

下面的程序仅用几行语句就可以使一个网页具有很强的交互性。

```
<html>
<head>
 <title>JavaScript Unleashed</title>
</head>
<body>
 <script type="text/javascript">
 <!--
  // 变量声明
  var payMoney;
  payMoney = 10.00;
  if(payMoney >
      document.write("非常感谢您的光临!");
  }else{
      document.writeln("谢谢,真希望您能再买点什么！");
  }
 // -->
 </script>
 </body>
 </html>
```

本例说明的是根据顾客购买商品的多少，售货员的两种不同态度。其执行结果如图 3-3 所示。

图3-14

3.11.2 循 环 语 句

在程序中使用循环有多种用途。一种非常简单但又非常普遍的用途就是用它来计数。例如，编写一个显示 0 到 9 的程序是非常容易的事情。您只需写 10 条命令来显示每一个数字：

```
document.writeln("0 ");
document.writeln("1 ");
…
document.writeln("9 ");
```

它的工作就是从 0 数到 9，但是，如果需要您数到 1000 又该怎么办呢？虽然您可以用同样的方法来实现，但是书写这些语句将花费您很多的时间。诸如此类的问题，JavaScript 都提供解决办法，即多种循环语句的使用。

1. For 语句

For 语句基本使用格式如下：

for（[initializing_expr]；[condition_expr]；[loop_expr]）{
Statements
}

其中，[initializing_expr]表示初始化表达式，[condition_expr]表示条件表达式，[loop_expr]表示循环增量。For 语句通常用初始化表达式声明一个变量作为循环计数器及初始值，接着只有条件表达式返回 true，才可能执行{}中的语句。每循环一次，循环计数器的值将以循环增量为步长进行增减。

下面的例子是用 For 语句将 0~99 中的每一个数字都显示在一个标准的页上，其中每显示 10 个数提行。

```
<html>
```

```
<head>
      <title>JavaScript Unleashed</title>
<body>
  <script type="text/javascript">
  <!--
    // 写一个标题
    document.write("显示从 0 到 99 的 100 个数字:");
    document.write('<hr size="0" width="50%" align="left">');
for (var i = 0; i < 100; ++i) {
      // 每显示 10 个数提行
      if(i%10 == 0) {
        document.write('<br>');
      }
      // 往标准页上写数字
      document.write(i + ",");
    }
 // -->
  </script>
</body>
</html>
```

从本例中可以看到：初始表达式首先声明变量 i 并赋值为 0，然后根据条件表达式判断 i 是否小于 100，此时 i = 0 小于 100，条件表达式返回值为 true，于是{}中的语句就被执行，并计算++i 的值使 i 的值加 1，这样就完成一次循环。接下来只要 i 小于 100 循环就将继续，直至 i=100 条件表达式返回值 false，循环终止。本例的执行结果如图 3-4 所示。

当然，如果条件表达式返回值 false，{}中的所有语句都不会执行。

和 if 语句一样，For 语句也可嵌套使用，下面的例子是用 For 语句的嵌套来产生一个 10×10 的网格坐标。

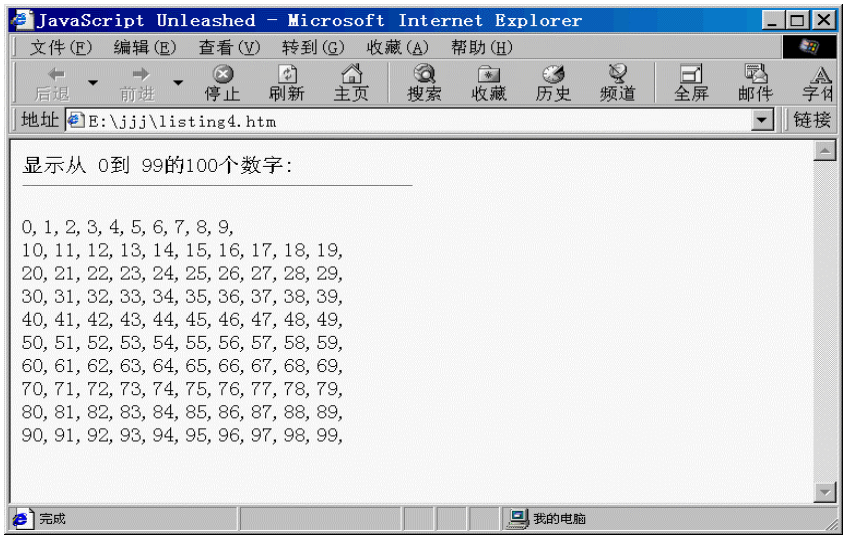


图3-15

```
<html>
<head>
  <title>JavaScript Unleashed</title>
</head>
<body>
  <script type="text/javascript">
  <!--
    document.write("显示所有（x,y）为 (0,0)到 (9,9)的坐标:<br>")
    for (var x = 0; x < 10; ++x) {
      for (var y = 0; y < 10; ++y) {
       document.write("(" + x + "," + y + "),");
      }
    document.write('<br>');
    }
  // -->
  </script>
</body>
</html>
```

本例的执行结果如图 3-5 所示。

在 For 语句的嵌套使用过程中，并不是只允许使用两层循环嵌套，您完全可以使用多重循环嵌套结构。

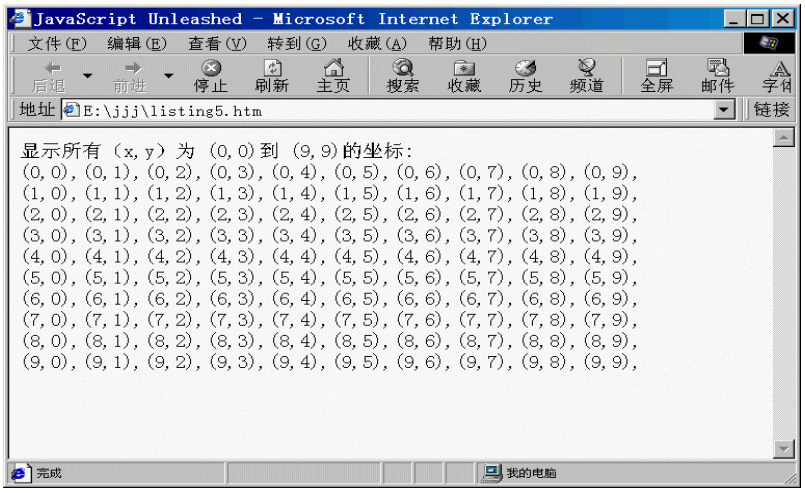


图3-16

2. For..in 语　句

要使用 For..in 循环语句，您需要先对 JavaScript 的对象有一个基本的了解。建议读者

先了解后面有关 JavaScript 对象的内容，再回来学习以下 For..in 循环语句。

利用 For..in 语句，可以根据对象的每一种不同的属性执行一系列语句。for..in 循环语句适用于 JavaScript 任何对象，不管这种对象有没有属性。由于每一种属性执行一次循环，因此如果该对象没有任何属性，则不进行任何循环。for..in 还可用于处理自定义对象。自定义 JavaScript 对象的变量被认为是它的一个属性，因此每一个自定义对象都要执行一次循环。For..in 语句的语法格式如下：

for（property in object）{
Statements
}

其中，property 表示对象的属性，object 表示对象。对于每一次循环，属性被赋以对象的下一个属性名一直到所有的属性名都使用了为止。下面的例子把 Doucment 对象的每一个属性名和它的属性值一起显示。

```
<html>
<head>
  <title>JavaScript Unleashed</title>
</head>
<body>
  <script language="JavaScript1.1" type="text/javascript">
  <!--
    //变量声明
    var anObject = document;
    var propertyInfo = "";
    for (var propertyName in anObject) {
      propertyInfo = propertyName + " = " + anObject[propertyName];
      document.write(propertyInfo + "<br>");
    }
 // -->
  </script>
</body>
</html>
```

本例的执行结果如图 3-6 所示。

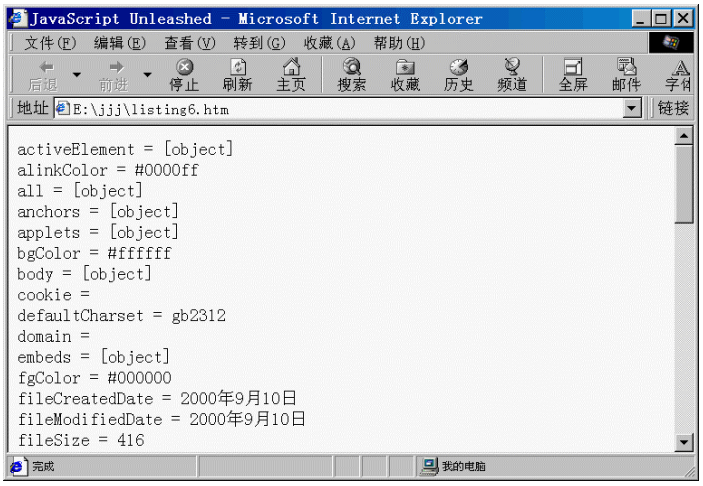


图3-17

for.in 循环要在 JavaScript 1.1 以上版本才能使用。它在 Microsoft Internet Explorer 3.0（Jscript 1.0）下无法正常运行。

3. while 语　句

While 语句的功能与 for 语句极为相似，只是它不能直接在其声明中初始化变量以及使变量增值，必须预先声明变量，并在语句块（Statements）中使其增量或减量。While 语句的语法结构如下所示：

While（condition_expr）{

Statements

}

当条件表达式（condition_expr）的返回值为真，则执行{}中的语句组（Statements），直到检测到条件表达式的返回值为假时，循环终止。

下面的例子是列出从 0 到 10 所有整数的和。

```
<html>
<head>
  <title>JavaScript Unleashed</title>
</head>
<body>
  <script type="text/javascript">
  <!--
    //变量声明
    var i = 0;
    var result = 0;
    var status = true;
    document.write("0");
    while(status){
      result += ++i;
      document.write(" + " + i);
      if(i == 10){
        status = false;
     }
    }
    document.writeln(" = " + result);
    // -->
  </script>
</body>
</html>
```

本例说明了如何应用逻辑变量作为判断循环是否继续进行的标志。这种状态变量要提前声明并使它的值为真。一旦 i 的值等于 10，状态变量值就被设置为假，使得循环终止。其执行结果如图 3-7 所示。

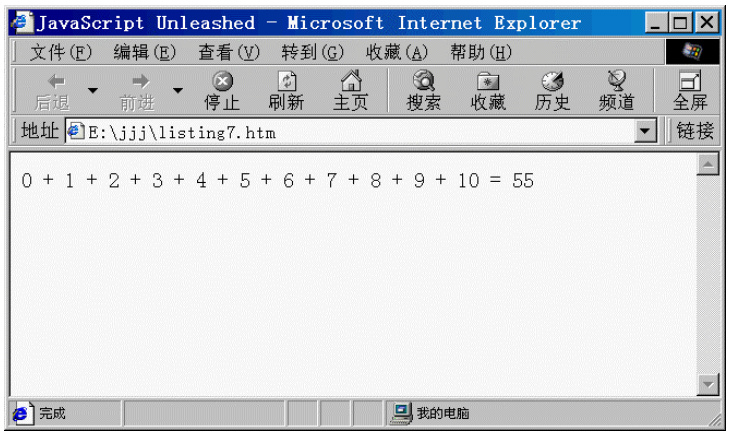


图3-18

4. Do..while 语　句

Do..while 结构从 JavaScript 1.2 版后开始提供。它的功能和 while 语句差不多，只不过它是在执行完第一次循环之后才检测条件表达式的值，这意味着包含在大括号中的程序段至少要被执行一次。

下面给出了一个例子，在一个标准页中显示从 1 到 10 的数字。

```
<html>
<head>
   <title>JavaScript Unleashed</title>
</head>
<body>
<script language="JavaScript1.2" type="text/javascript">
   <!--
      // 变量声明
      var userEntry =10;
      var x = 1;
      do{
         document.writeln(x);
         x++;
      }while(x <= userEntry);
   // -->
   </script>
</body>
</html>
```

本例的执行结果如图 3-8 所示。

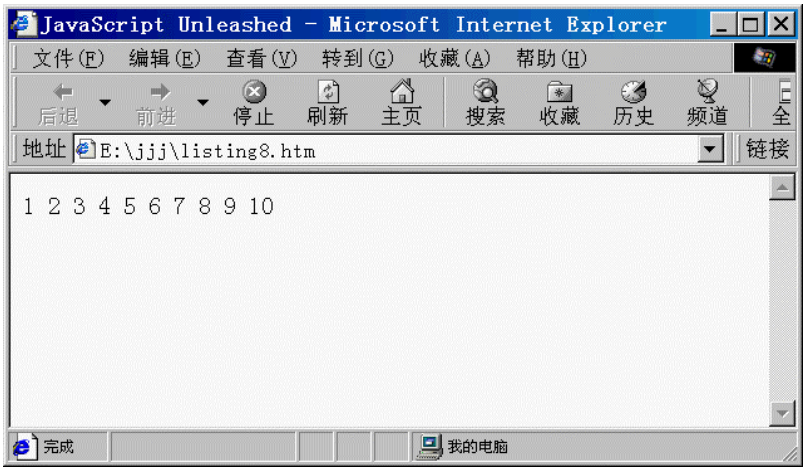


图3-19

5. Break 和 Continue 语句

在使用循环语句时，我们要注意在默认情况下，循环将无休止地进行，除非循环条件表达式的值为假。有时，您可能想提前中断或跳过循环，要实现这一点，您只需在循环语句块中添加 Break 或 continue 语句就可以了。

Break 语句中断所有循环，而 continue 语句则跳过本次循环的剩余语句，然后开始下一次循环。您可以从下面的程序中看出二者的区别，该程序用最基本的方法来求一个数 n 的近似平方根。

```
<html>
<head>
  <title>JavaScript Unleashed</title>
</head>
<body>
  <script type="text/javascript">
  <!--
    // 变量声明
    var highestNum = 0;
    var n = 175;
    for(var i = 0; i < n; ++i){
      document.write(i + "<br>");
      if(n < 0){
        document.write("n 不能为负数。");
        break;
      }
      if (i * i <= n) {
        highestNum = i;
        continue;
      }
      document.write("<br>完成!");
      break;
    }
```

```
        document.write("<br>"+ "小于或等于"+ n +"的平方根的整数"+" = " + highestNum);
    // -->
    </script>
</body>
</html>
```

由本例可以看出，程序一开始使 i 的值为 0，并在 for 循环开始时显示 i 的值，接着，程序检测确认 n 为非负数。如果 n 为负数，就终止（Break）循环，程序将跳出该 for 循环，去续执行大括号后面的语句。如果 n 为正数，则 i 与自身相乘，并将结果与 n 比较，如果 i 小于或等于 n，则将此时 i 的值保存到变量 highestNum 中，接下来的 Continue 语句就跳过本次 for 循环的剩余语句，然后循环又从头开始，当 i 的平方值大于 n，程序就跳过 Continue 语句达到 Break 语句，该语句完全终止循环。该程序的执行结果如图 3-9 所示。

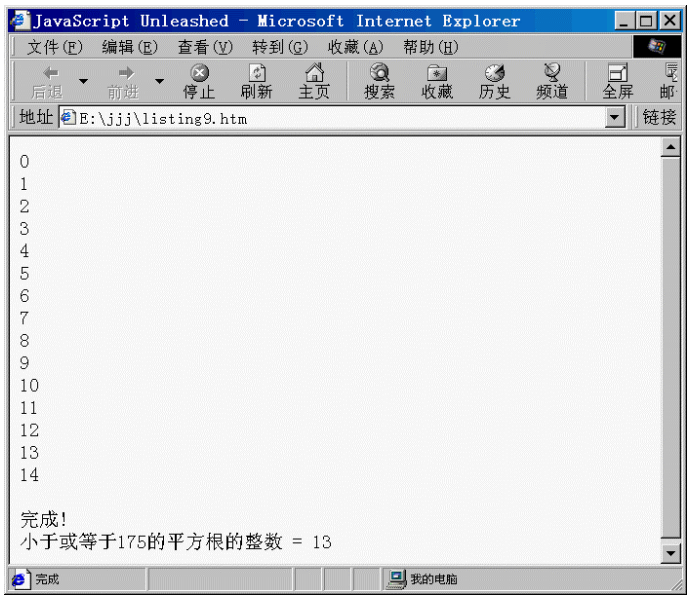


图3-20

3.11.3 标 签 语 句

随着 JavaScript 1.2 版本的出台，该语言提供了一种使使用 Continue 和 Break 语句更为有效的办法。标签语句可以放置在任何控制结构之前，这些控制结构能够嵌套其它语句。它可以使您跳出条件语句或循环语句，而转到您程序中的其它具体位置。在下面的程序中您可以看到该语句的用法。

```
<html>
<head>
    <title>JavaScript Unleashed</title>
</head>
<body>
    <script language="JavaScript1.2" type="text/javascript">
    <!--
```

```
    // 变量声明
    var stopX = 4;
    var stopY = 9;
    document.write("所有在(0,0)和(");
    document.write(stopX + "," + stopY + ")的 x,y 组合有:<br>");

  loopX:
    for(var x = 0; x < 10; ++x){
      for(var y = 0; y < 10; ++y){
        document.write("("+x + "," + y +") ");
        if((x == stopX) && (y == stopY)){
          break loopX;
        }
      }
      document.write("<br>");
    }
    document.write("<br>循环完成后 x=" + x);
    document.writeln("<br>循环完成后 y=" + y);
  // -->
  </script>
</body>
</html>
```

在本例中，for 循环被贴上了用户自定义 loopX 标签，不管程序如何嵌套，它都能跳出或继续 for 循环。

如果没有标签语句，Break 语句就只终止产生 y 值的循环。图 3-10 分别显示了 x 和 y 的值达到 4 和 9 时的结果。

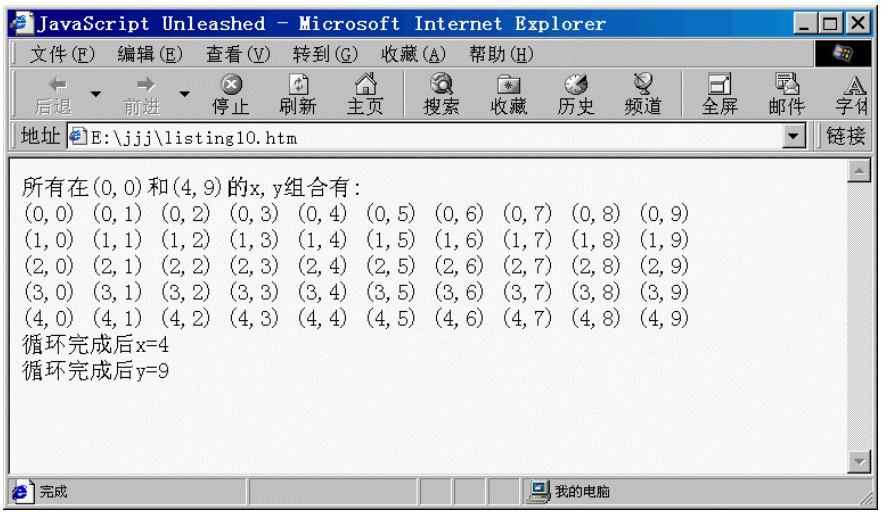


图3-21

3.11.4 With 语　句

有了 With 语句，在存取对象属性和方法时就不用重复指定参考对象。在 With 语句块

中，凡是 JavaScript 不识别的属性和方法都和该语句块指定的对象有关。With 语句的语法格式如下所示：

With（Object）{

Statements

}

对象指明了当语句组中对象缺省时的参考对象。这里我们用较为熟悉的 Document 对象对 With 语句举例。

例如：当使用与 Document 对象有关的 write( )或 writeln( )方法时，往往使用如下形式：

document.writeln(“Hello!”)

如果需要显示大量数据时，就会多次使用同样的 document.writeln（）语句。这时就可以像下面的程序那样，把所有以 Document 对象为参考对象的语句放到 With 语句块中，从而达到减少语句量的目的。下面是一个 With 语句使用的例子。

```
<html>
<head>
  <title>JavaScript Unleashed</title>
</head>
<body>
  <script type="text/javascript">
  <!—

    with(document){
      write("您好!");
      write("<br>这个文档的标题是: \"" + title + "\".");
      write("<br>这个文档的 URL 是: " + URL);
      write("<br>现在您不用每次都写出 document 对象的前缀了!");
    }

  // -->
  </script>
</body>
</html>
```

这样，您在使用 document 的方法和属性时就可以去掉 Document 前缀。

请注意程序中的 title 和 URL 均是 Document 对象的属性，一般情况下应写作 document.title 和 document.URL。使用 With 语句，您只需指定一次参考对象，这同把每一行都用 document.writeln（）打印下来的结果一样。这个例子的执行结果如图 3-10 所示。

由于浏览器的不同，在本例中您可能看到 URL 的一种编码格式。

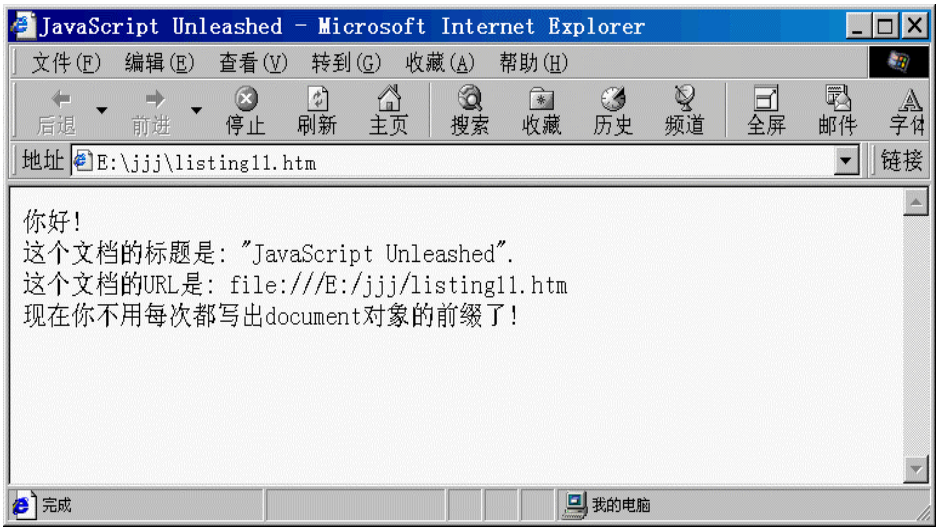


图3-22

3.11.5 Switch 语　句

Switch 语句用于将一个值同许多其它值比较，并按比较结果选择下面该执行哪些语句。Switch 语句的使用格式如下：

```
Switch( value ){
    Case value1:
        Statements
        Break;
      Case value2:
        Statements
        Break;
    …
    Default:
        Statements
        Break;
}
```

它表明如果 Switch 语句中的变量 value 与 Case 语句中哪一个值（value1、value2、…）相等，就执行那个 Case 语句后的语句块，直到遇到 break 语句为止。如果没有一个 Case 语句中的值与 Switch 语句中的变量 value 值相等，则将执行 Default 语句后面的语句块。Break 语句用来制止进一步执行 Switch 语句中剩下的其它任何语句。如果没用 Break 语句，不管 request 与那种情况是否匹配，每一种情况后面剩下的语句都将被执行。

您可能认为这只需要使用多个 if 语句就可以实现，但过多地使用 if 语句会降低程序的可读性，程序容易混乱。这时使用 Switch 语句是一种很好的途径。Switch 语句不仅可读性更强，还允许您为找不到匹配时指定默认的一系列语句。下面就是一个 Switch 语句使用的典型例子：

```
<html>
<head>
  <title>JavaScript Unleashed</title>
```

```
</head>
<body>
  <script language="JavaScript1.2" type="text/javascript">
  <!--
    // 变量声明
    var request = "Name";

    switch(request){
      case "Logo" :
        document.write('<img src="logo.gif" alt="Logo">');
        document.write("<br>");
        break;
      case "Name" :
        document.write("Software Inc.");
        document.write("<br>");
        break;
      case "Products" :
        document.write("MyEditor");
        document.write("<br>");
        break;
      default:
        document.write("www.mysite.com");
        break;
    }
  //-->
  </script>
</body>
</html>
```

在本例中 request 变量是可变的，这取决于用户的需要，如果 request 被赋值为 Names，用该值同每种情况比较，第二种情况与 request 相等，则执行它后面的语句。本例执行的结果如图 3-12 所示。

图3-23

注意 只有 JavaScript 1.2 以上版本才有 Switch 语句。

第4章 Window（窗口）对象

主要内容

- 窗口对象的属性
- 窗口对象的方法
- 窗口对象的事件
- 创建和关闭窗口

本章导读

在浏览器中，Window（窗口）对象是所有对象的 Parent（根）。在本章中我们将分别介绍 Window（窗口）对象的属性、方法、事件。

4.12 Window（窗口）对象的属性

在浏览器中的 Window（窗口）对象主要有如下的属性：Name，Length，Parent，Self，Top，Status，Default Status，Opener，Closed。

在下面我们将主要介绍 Window（窗口）对象中最常用的属性，首先让我们来看看 Default Status 和 Status 属性。

Default Status 和 Status 属性是在窗口中的最下面的状态条上按照事件发生的顺序显示状态信息。当然我们不能将 Status 和 Default Status 弄混淆了，Default Status 是在状态条上显示缺省值的信息，Status 是用户自定义状态显示信息。

在下面的实例中，我们能更充分了解 Default Status 和 Status 属性是在窗口中使用的。首先在状态栏中显示缺省信息，当从下拉菜单中选择一项时，则状态栏中的信息发生相应地改变，从实例中我们可以很清楚地了解到怎样改变状态栏的显示信息。

```
<html>
<head>
<title>状态信息</title>
<SCRIPT LANGUAGE="JavaScript">
<!--
window.defaultStatus = "欢迎访问联谊主页"
function changeStatus()
{
window.status = "单击这里进入娱乐焦点主页"
}
function changeDefaultStatus()
{
window.defaultStatus = window.document.statusForm.messageList.
options[window.document.statusForm.messageList.
selectedIndex].text
}
//-->
</SCRIPT>
</head>
<body>
<p> </p>
<p> </p>
<p align=center>
<font color="#008040">
<font size=7>
```

```
<strong>http://www.ly.scit.edu.cn</strong></font></font></p>
<p align=center>
<a href="http://www.ly.scit.edu.cn" onMouseOver="changeStatus();
return true">Go...</a></p>
<form name="statusForm" method="POST">
<p><br>
<br>
<br>
<br>
</p>
<p align=center>
<font size=1>从下拉菜单中选择相应的选项，则状态栏信息发生改变</font></p>
<p align=center><select
name="messageList"
size=1>
<option selected>欢迎访问联谊</option>
<option>联谊娱乐焦点</option>
<option>联谊软件仓库</option>
<option>联谊影院广场</option>
</select>
<input
type=button
name="Change"
value="Change"
onClick="changeDefaultStatus()"></p>
</form>
</body>
</html>
```

下面的代码是改变状态栏信息的函数 ChangeDefaultStatus()。

```
function changeDefaultStatus()
{
window.defaultStatus = window.document.statusForm.messageList.
options[window.document.statusForm.messageList.
selectedIndex].text
}
```

由代码 onClick="changeDefaultStatus()"调用函数来改变状态栏的信息。

在浏览器中预览实例的效果如图 4-1 所示。

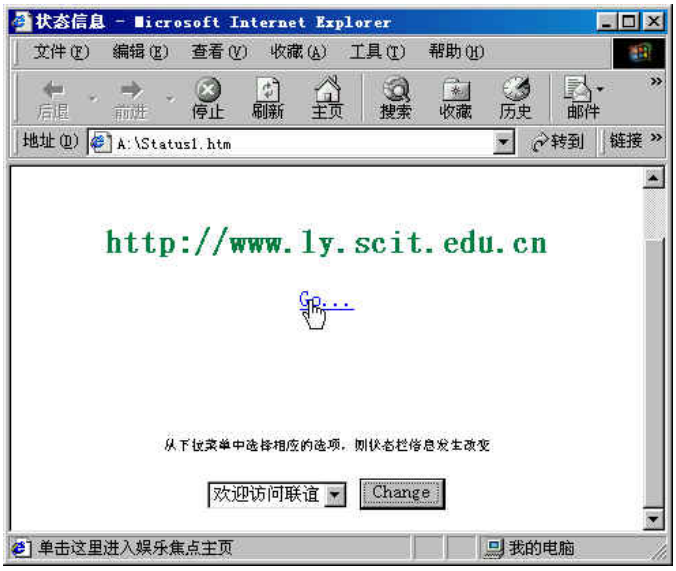


图4-24

4.13 Window（窗口）对象的方法

在浏览器中的 Window（窗口）对象主要有如下的方法：Item，alert，blur，close，confirm，open，focus。

在下面我们将主要介绍 Window（窗口）对象中最常用的方法，在下面将介绍 open（）方法。

4.13.1 Open（）方　法

使用 Window（窗口）对象的 Open（）方法来打开一个新的浏览器窗口，打开新窗口的参数是通过 URL 传递给 Open（）的。

在下面的实例中充分利用了 Window 的 Open（）方法。该实例将在新打开窗口的同时，定义新打开窗口的大小，并且在新打开的窗口中加入状态行和其它选项。下面是实例程序的代码清单。

Openwindow.htm

```
<HTML>
<TITLE>
新建窗口实例
</TITLE>
<BODY>
<FORM>
<CENTER>
<BR>
<H1>打开一个新的窗口实例……</H1>
<BR>
<BR>
```

```
<INPUT TYPE=BUTTON Value=”新建窗口” onclick=”OpenWindow()”>
</CENTER>
</FORM>
</BODY>
<SCRIPT LANGUAGE=JavaScript>
Function OpenWindow()
{
   Window.open(“NewWindow.htm”,null,”height=300,width=300,status=yes,toolbar=no,menuba=no,location=no”)
}
</SCRIPT>
</HTML>

NewWindow.htm
<HTML>
<HEAD>
<TITLE>
新建窗口
</TITLE>
<BODY>
<CENTER>
<H1>
新打开的窗口
</H1>
</CENTER>
<FORM>
<CENTER>
<INPUT TYPE=BUTTON Value=”关闭窗口” OnClick=”winclose()”>
</CENTER>
</FORM>
</BODY>
<SCRIPT LANGUGE=JavaScript>
function winclose()
{
   Window.close()
}
</SCRIPT>
</HTML>
```

下面的代码是调用 Window open（）方法来打开一个新的浏览器窗口，并且设置打开新建窗口的窗体属性。

```
<SCRIPT LANGUAGE=JavaScript>
Function OpenWindow()
{
   Window.open(“NewWindow.htm”,null,”height=300,width=300,status=yes,toolbar=no,menuba=no,location=no”)
```

```
}
</SCRIPT>
```

下面的代码是调用 Window.close（）方法来关闭新打开的浏览器窗口。

```
<SCRIPT LANGUGE=JavaScript>
function winclose()
{
   Window.close()
}
</SCRIPT>
```

在浏览器中预览的效果如图 4-2 所示。

图4-25

当单击“新建窗口”按钮时，打开一个新的浏览器窗口，并且新建的窗口都是按照我们设置的属性参数显示出来。如图 4-3 所示。

图4-26

4.13.2 alert（）方　法

使用 Window（窗口）对象的 alert（）方法，可以在屏幕中显示一个警告框，这些警告框是用来显示一条简短的信息，向用户表明某个情况的发生，单击【OK】按钮来关闭警告框。

在下面的实例中我们将在页面中创建一个按钮，当用鼠标单击按钮时，将弹出一个警告框来显示某些警告信息。然后，当用户单击警告框中的按钮时，警告框将自动消失。下面是实例程序的代码清单。

Alert.htm

```
<HTML>
<HEAD>
<TITLE>
警告框实例
</TITLE>
</HEAD>
<BODY>
<FORM>
<CENTER>
<H1>
单击按钮
</H1>
<BR>
<BR>
<INPUT TYPE=BUTTON VALUE=”警告框” onClick=”alertwin()”>
</CENTER>
</FORM>
<SCRIPT LANGUAGE=JavaScript>
function alertwin()
{
   window.alert(“您将看到一个警告框！”)
}
</SCRIPT>
</BODY>
</HTML>
```

下面的代码是当用户单击按钮后，利用 Window.alert()方法来显示一个警告框。

```
<SCRIPT LANGUAGE=JavaScript>
function alertwin()
```

```
{
   window.alert("您将看到一个警告框！")
}
</SCRIPT>
```

在打开页面并且单击按钮后得到的页面显示效果如图 4-4 所示。

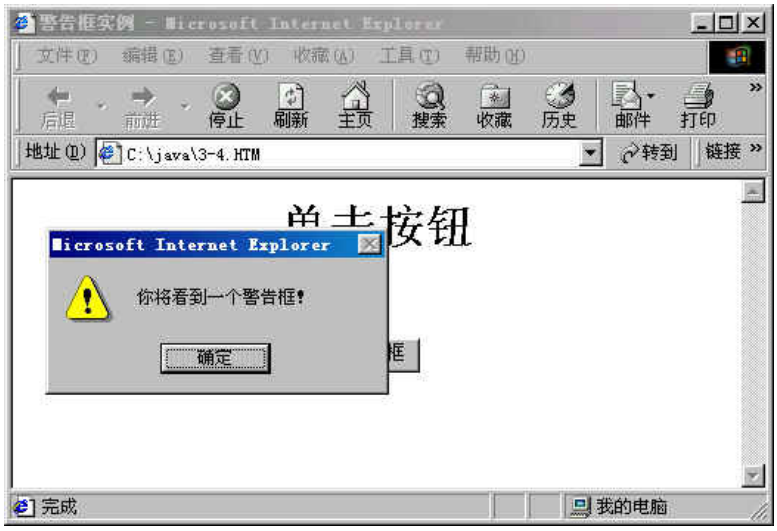


图4-27

4.13.3 Confirm（）方　法

使用 Window（窗口）对象的 confirm（）方法来显示一个可以接收用户输入的确认框，也就是当用户单击【OK】或【Cancel】按钮时，我们可以检测到哪一个动作发生了。

在下面的实例中我们将利用 Window（窗口）对象的 confirm（）方法打开一个带有信息的确认框，当用户单击【OK】或【Cancel】按钮时，关闭确认框，并且在窗体中显示一条信息来确认用户是单击的【OK】按钮还是【Cancel】按钮。下面是实例的代码清单。

Confimwin.htm

```
<HTML>
<HEAD>
<TITLE>
确认对话框实例
</TITLE>
</HEAD>
<BODY>
<FORM name=conform>
<CENTER>
<H1>
单击按钮
</H1>
<BR>
<INPUT TYPE=BUTTON VALUE="单击" onClick="conwin()">
```

```
<BR>
<BR>
<INPUT TYPE=TEXT NAME=BOX1 SIZE=20>
</CENTER>
</FORM>
</BODY>
<SCRIPT LANGUAGE=JavaScript>
function conwin()
{
if (confirm("单击【OK】按钮或者【CANCEL】按钮…"))
{
document.conform.box1.value="【OK】按钮被单击！"
}
else
{
document.conform.box1.value="【CANCEL】按钮被单击！"
}
}
</SCRIPT>
</HTML>
```

下面的代码是当用户单击按钮时，利用 Window.confirm()方法来显示一个确认框，如果用户单击确认框中的【OK】按钮，则该方法返回是真，否则返回假，在程序中通过判断返回的值，然后在文本框中显示某个按钮。

```
<SCRIPT LANGUAGE=JavaScript>
function conwin()
{
if (confirm("单击【OK】按钮或者【CANCEL】按钮…"))
{
document.conform.box1.value="【OK】按钮被单击！"
}
else
{
document.conform.box1.value="【CANCEL】按钮被单击！"
}
}
</SCRIPT>
```

在浏览器中打开页面并且单击按钮得到的效果如图 4-5 所示。

图4-28

当用户单击确认框中的【确认】按钮后得到的效果如图 4-6 所示。

图4-29

4.13.4 Show Modal Dialog（）方　法

使用 Window（窗口）对象的 Show Modal Dialog（）方法来创建对话框，这种方法是通过 dialog 对象来创建一个新的窗口。在 Internet Explorer 中的 dialog 对象的属性中有 width，height，dialogargs 和 return Value，但它只有一个方法——close（）方法。

下面的实例就是利用 Internet Explorer 的 Dialog 对象来显示如何使用对话框，在页面中显示一个按钮和一个文本框，当用户单击按钮时，将显示一个带有 7 个单选按钮，它们分别代表 7 种颜色的对话框，要求用户在 7 个单选按钮中选择一个单选按钮，然后单击【确认】按钮，在关闭对话框后同时在主页面窗口中显示用户作出的选择结果。

下面是该实例的代码程序清单。

Color.htm

```
<HTML>
<TITLE>
对话框实例
</TITLE>
<BODY>
<CENTER>
<FORM NAME=form1>
<BR>
<H1>
单击按钮，选择您喜欢的颜色
</H1>
<BR>
<INPUT TYPE=BUTTON Value=”选择颜色” onClick=”Selectcolor”>
<BR>
<BR>
<INPUT TYPE=TEXT NAME=”Textbox” SIZE=20>
<BR>
</FORM>
</CENTER>
</BODY>
<SCRIPT LANGUAGE=JavaScript>
function Selectcolor()
{
   document.form1.Textbox.value=Window.showModalDialog(“dlgwin.htm”)
}
</SCRIPT>
</HTML>
```

dlgwin.htm

```
<HTML>
<BODY>
<CENTER>
<FORM NAME= “form1”>
<BR>
<BR>
<H1>
选择您喜欢的颜色
</H1>
<BR>
<TABLE BORDER BGCOLOR=CYAN WIDTH=200>
<TR>
<TD>
<INPUT TYPE=RADIO NAME=RadioButtons onClick=radio1Clicked()>红
</TD>
</TR>
<TR>
<TD>
```

```
<INPUT TYPE=RADIO NAME=RadioButtons onClick=radio2Clicked()>橙
</TD>
</TR>
<TR>
<TD>
<INPUT TYPE=RADIO NAME=RadioButtons onClick=radio3Clicked()>黄
</TD>
</TR>
<TR>
<TD>
<INPUT TYPE=RADIO NAME=RadioButtons onClick=radio4Clicked()>绿
</TD>
</TR>
<TR>
<TD>
<INPUT TYPE=RADIO NAME=RadioButtons onClick=radio5Clicked()>青
</TD>
</TR>
<TR>
<TD>
<INPUT TYPE=RADIO NAME=RadioButtons onClick=radio6Clicked()>蓝
</TD>
</TR>
<TR>
<TD>
<INPUT TYPE=RADIO NAME=RadioButtons onClick=radio7Clicked()>紫
</TD>
</TR>
</TABLE>
<BR>
<INPUT TYPE=BUTTON Value=”确认” onClick=”OKBbutton()”>
<INPUT TYPE=BUTTON Value=”取消” onClick=”CancelButton()”>
</FORM>
</CENTER>
</BODY>
<SCRIPT LANGUAGE=JavaScript>
var seceledcolor=”没有选择”
function radio2Clicked()
{
   seceledcolor=”红”
}
function radio1Clicked()
{
   seceledcolor=”橙”
}
function radio1Clicked()
{
   seceledcolor=”黄”
```

```
}
function radio1Clicked()
{
   seceledcolor=”绿”
}
function radio1Clicked()
{
   seceledcolor=”青”
}
function radio1Clicked()
{
   seceledcolor=”蓝”
}
function radio1Clicked()
{
   seceledcolor=”紫”
}
function OKButton()
{
   window.returnValue=”您选择的颜色为：”+seceledcolor
   window.close()
}
function CancelButton()
{
window.returnValue=”您没有作出选择！”
   window.close
}
</SCRIPT>
</HTML>
```

下面的代码是利用 window（窗口）对象的 ShowModalDialog（）方法来创建一个对话框，并且同时在创建的对话框中载入新的文档“dlgwin.htm”，在创建的对话框中显示 7 个单选按钮，要求用户作出选择。

```
<SCRIPT LANGUAGE=JavaScript>
function Selectcolor()
{
   document.form1.Textbox.value=Window.showModalDialog(“dlgwin.htm”)
}
</SCRIPT>
```

下面的代码是 7 个单选按钮的响应函数，当用户单击某个单选按钮时，就将选择的对应按钮代表的颜色值传递给变量“seceledcolor”。

```
<SCRIPT LANGUAGE=JavaScript>
var seceledcolor=”没有选择”
function radio1Clicked()
{
   seceledcolor=”红”
}
function radio2Clicked()
```

```
{
   seceledcolor=”橙”
}
function radio3Clicked()
{
   seceledcolor=”黄”
}
function radio4Clicked()
{
   seceledcolor=”绿”
}
function radio5Clicked()
{
   seceledcolor=”青”
}
function radio6Clicked()
{
   seceledcolor=”蓝”
}
function radio7Clicked()
{
   seceledcolor=”紫”
}
```

下面的代码是用户单击【确定】按钮时，通过 showModalDialog()方法的返回值将用户的选择颜色传递给主窗口，并且将对话框窗口关闭。

```
function OKButton()
{
   window.returnValue=”您选择的颜色为：”+seceledcolor
   window.close()
}
function CancelButton()
{
window.returnValue=”您没有作出选择！”
   window.close（）
}
```

在浏览器中打开页面时并且单击按钮得到的效果如图 4-7 所示。

当用户单击【选择颜色】按钮时弹出的选择对话框，如图 4-8 所示。

图4-30

图 4-8

图 4-9

当用户选择了对话框中的某个单选按钮，并且单击【确定】按钮关闭对话框后回到主窗口显示作出的选择颜色，如图 4-9 所示。

4.13.5 Focus（）方　法

使用 windows.focus（）方法获取焦点位置。当用户在多窗口间切换时，使用 window.focus（）方法可以将要引用的窗口送到所有窗口的前面。

这种方法成功的关键是确保您对目标窗口的引用是正确的。因此当一个窗口需要把某个窗口送回到焦点位置，一定要为每个窗口赋予一个惟一标识的名字，来确定将某个窗口传递给焦点位置。

下面的实例就是使用 window.focus（）方法及其反向操作 window.blur（）方法。在文件中创建了一个双窗口环境，在任何一个窗口中都可以将另外一个窗口提前到当前窗口前面。下面是实例程序的清单。

```
<HTML>
<HEAD>
<TITLE>焦点定位</TITLE>
<SCRIPT LANGUAGE="JavaScript">
```

```
var newWindow = null
function makeNewWindow()
{
   if (!newWindow || newWindow.closed)
   {
   newWindow = window.open("","","width=150,height=150")
   var newContent = "<HTML><HEAD><TITLE>新建一个窗口</TITLE></HEAD>"
   newContent += "<BODY bgColor='blue'><H1>一个新的窗体测试</H1>"
   newContent += "<FORM><INPUT TYPE='button' VALUE='切换一个新的窗体'
onClick='self.opener.focus()'><BR><BR>"
   newContent += "<FORM><INPUT TYPE='button' VALUE='返回上一窗体' onClick='self.blur()'>"
   newContent += "</FORM></BODY></HTML>"
   newWindow.document.write(newContent)
   newWindow.document.close()
   }
else
{
   newWindow.focus()
   }
}
</SCRIPT>
</HEAD>
<BODY>
<FORM>
<INPUT TYPE="button" NAME="newOne" VALUE="打开新的窗体" onClick="makeNewWindow()">
</FORM>
</BODY>
</HTML>
```

下面的代码程序是使用 JavaScript 来创建一个新的窗体，在新的窗体中创建了两个按钮，分别用来切换窗体和返回上一个窗体。分别使用 self.opener（）来指向新窗口的引用，而使用 self.blur（）方法来指向子窗口本身。

```
<SCRIPT LANGUAGE="JavaScript">
var newWindow = null
function makeNewWindow()
{
   if (!newWindow || newWindow.closed)
   {
   newWindow = window.open("","","width=150,height=150")
   var newContent = "<HTML><HEAD><TITLE>新建一个窗口</TITLE></HEAD>"
   newContent += "<BODY bgColor='blue'><H1>一个新的窗体测试</H1>"
   newContent += "<FORM><INPUT TYPE='button' VALUE='切换一个新的窗体'
onClick='self.opener.focus()'><BR><BR>"
   newContent += "<FORM><INPUT TYPE='button' VALUE='返回上一窗体' onClick='self.blur()'>"
   newContent += "</FORM></BODY></HTML>"
   newWindow.document.write(newContent)
   newWindow.document.close()
```

```
    }
else
{
    newWindow.focus()
    }
}
</SCRIPT>
```

在浏览器中预览窗体得到的效果如图 4-10 所示。

下面是单击按钮后打开一个新的窗体，如图 4-11 所示。

图 4-10

图 4-11

4.14 创建和关闭窗口

我们在前面已经介绍了窗口对象的属性、方法、事件。在下面我们将使用 JavaScript 来打开和关闭一个浏览器窗口。作为一个开发者可以通过调入一个文档来创建一个窗口，在创建窗口时，还可以定义窗口的一些特性，例如：定义窗口的尺寸、位置等信息。

4.14.1 创 建 窗 口

使用 JavaScript 来创建一个窗口的 HTML 语法如下：

windowVar=window.open(URL,windowName,[,windowFeature])

使用 window 对象的 open（）方法的参数如下：

- **URL**：URL 是指向的一个目标窗口。也就是在某个浏览器窗口中创建这个新的窗口。如果 URL 这个参数为空，那么浏览器将创建一个空白的窗口。在创建窗口时也可以使用 write（）方法来创建动态的 HTML 语句以便创建新窗口；

- **Window Name**：是创建的窗口对象的名字。赋予窗口对象一个名字是为了以后通过调用该名字来访问该窗口对象。当然对这个参数并非是一定需要的，也可以不赋予窗口名字。

如果使用 window 对象的 open（）方法创建窗口成功的话，那么将返回一个窗口对象的句柄，如果没有创建成功将返回一个空值。

4.14.2 定 义 窗 口

在 JavaScript 应用程序中可使用单个或多个框架结构，此时可能需要某些方法去定义窗口。在 JavaScript 中提供了 4 种方法来定义窗口。

其实每种方法都是通过定义窗口的属性来完成的。如果需要了解更多的信息，可以参考窗口对象的属性以及框架对象的相关内容。

4.14.3 关 闭 窗 口

可以使用 window 对象的 close（）方法来关闭一个窗口。假如关闭的是当前的窗口，那么我们只需简单地调用 window.close（）方法来关闭窗口即可。如果关闭的不是当前窗口对象，如 alert（）、setTime（）等窗口对象时，使用 window.close（）方法就必须有一个目标对象的参考。如果调用它自身的 close（）方法就比调用其它对象的方法来关闭更可靠。因为这些对象本身都有自己的 close（）方法。

使用一个函数或者事件的方法来关闭窗口对象，都需要如下的语法来定义：

```
<form>
<INPUT TYPE=”BUTTON” VALUE=”Close window” onClick=”top.close()” >
</form>
```

在关闭一个窗口对象时，一个比较好的方法是在窗口对象中提供一个按钮或者是一个连接来关闭窗口对象。特别是对于新用户来说，他们可以比较清楚地了解到什么时候可以将窗口关闭，而不影响其它工作。

在下面的实例中将根据用户选择的某些属性来创建一个窗口。当用户做出选择后，单击 open window 按钮将打开一个新窗口。下面是实例程序的代码清单：

```
<html>

<head>
<title>Window Open</title>
<script LANGUAGE="JavaScript">
<!--
        var newWindow
         function openWindow() {
                var winAtts = ""
               if (document.winOptions.toolbarOption.checked) {
                      winAtts += ",toolbar=1" }
               if (document.winOptions.menubarOption.checked) {
                      winAtts += ",menubar=1" }
```

```
            if (document.winOptions.scrollbarsOption.checked) {
                winAtts += ",scrollbars=1" }
            if (document.winOptions.resizableOption.checked) {
                winAtts += ",resizable=1" }
            if (document.winOptions.statusOption.checked) {
                winAtts += ",status=1" }
            if (document.winOptions.locationOption. checked) {
                winAtts += ",location=1" }
            if (document.winOptions.directoriesOption.checked) {
                winAtts += ",directories=1" }
            if (document.winOptions.copyHistoryOption.checked) {
                winAtts += ",copyhistory=1" }
            if (document.winOptions.customSizeOption.checked)
                                {
              winAtts+= ",height="+document.winOptions.heightBox.value;
              winAtts+=",width=" + document.winOptions.widtBox.value;

                }

            if (document.winOptions.pageType[1].checked) {
                var urlVar = ""
                urlVar = document.winOptions.urlBox.value
                newWindow = window.open(urlVar,"newWindow",winAtts) }
            else {
                newWindow = window.open("","newWindow",winAtts)
                newWindow.document.write("<H2>打开新的窗口</H2><p>")
            }
        }
        // Close Window
        function closeWindow() {
            newWindow.close()
        }
// -->
</script>
</head>

<body background="../lt_rock.gif">

<h1><font color="#008040">Window Open Example</font></h1>

<p><b><i>请先选择一些窗口属性然后单击&amp;quot;新建窗口&amp;quot;按钮. </i></b></p>

<form name="winOptions" method="POST">
  <p>你将创建什么样式的窗口</p>
  <p><input type="radio" checked name="pageType" value="existing">已经存在的网页 <input
  type="text" size="30" maxlength="256" name="urlBox"></p>
  <p><input type="radio" name="pageType" value="dynamic">动态网页</p>
  <hr>
  <p>窗口属性:</p>
```

```
<pre><input type="checkbox" name="toolbarOption" value="ON">Toolbar <input
type="checkbox" name="menubarOption" value="ON">Menubar <input type="checkbox"
name="scrollbarsOption" value="ON">Scrollbars <input type="checkbox"
name="resizableOption" value="ON">Resizable</pre>
<pre><input type="checkbox" name="statusOption" value="ON">Status <input
type="checkbox" name="locationOption" value="ON">Location <input type="checkbox"
name="directoriesOption" value="ON">Directories <input type="checkbox"
name="copyHistoryOption" value="ON">Copy History</pre>
<pre><input type="checkbox" name="customSizeOption" value="ON">Custom Size</pre>
<pre>Width: <input type="text" size="5" maxlength="5" name="widtBox"> Height: <input
type="text" size="5" maxlength="5" name="heightBox"> <input type="button"
name="OpenButton" value="新建窗口" onClick="openWindow()"> <input type="button"
name="CloseButton" value="关闭窗口" onClick="closeWindow()"></pre>
</form>

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

实例在浏览器中预览的效果如图 4-12 所示。

图 4-12

第5章 document对象

主要内容

- document 对象的属性
- document 对象的方法

本章导读

在浏览器中，与用户进行数据交换都是通过客户端的 JavaScript 代码来实现的，而完成这些交互工作大多数是 document 对象及其部件进行的，因此 document 对象是一个比较重要的对象。我们将在本章中详细介绍 document 对象。

5.15 document 对象的属性

document 对象主要有如下的属性：alinkcolor，bgcolor，cookie，domain，embeds，fgcolor，o,ages，layers，linkcolor，location，title，url，vlinkcolor。

在下面我们将介绍如何使用 document 对象的一些常用属性来设置 Web 页面的某些特性，比如可以设置文本的颜色、背景色、链接颜色、标题颜色等信息。

5.15.1 BgColor 属　性

下面的代码实例就是通过改变 document 的属性 bgcolor（背景颜色）来改变页面的背景色。实例程序代码的清单如下：

bgcolor.htm

```
<HTML>
<HEAD>
<TITLE>设置背景色实例</TITLE>
</HEAD>
<BODY>
<CENTER>
<FORM>
<BR>
<H1>
利用 document's bgColor 属性
<BR>
<BR>
来改变页面的背景色.....
</H1>
<BR>
<BR>
<BR>
<INPUT TYPE = BUTTON Value = "随机改变背景色" onClick = "colorBackground()">
</FORM>
</CENTER>
</BODY>
<SCRIPT LANGUAGE = JavaScript>
  function colorBackground()
  {
      document.bgColor +=0X000008
  }
</SCRIPT>
</HTML>
```

下面的代码就是随机改变的页面背景色。

```
<SCRIPT LANGUAGE = JavaScript>
  function colorBackground()
  {
      document.bgColor +=0X000008
  }
</SCRIPT>
```

在浏览器中预览的页面效果如图 5-1 所示，当用户单击页面中的按钮时，页面的背景色将随机改变。

图5-31

5.15.2 Anchors 属　性

document.anchors 属性是以复数的形式提交一个关于文档中 anchor 对象的索引数组，通过数组可以精确地找到指定的 anchor 对象，并取回任何需要的 anchor 对象属性。

对 anchor 对象的引用是通过 document.anchors[i]来定位的，而且还可以通过检查 document.anchors.length 属性来确定数组中有多少个元素。

下面的实例就是利用 document.anchors 的属性来取得文档中的 anchors 的数目。页面动态地写出在文档中找到的 anchor 数目。并且根据找到的 anchor 对象，单击相应的按钮可以返回到相应的位置。实例程序的代码清单如下。

```
<HTML>
<HEAD>
<TITLE>document.anchors 属性实例</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function goNextAnchor(where)
{
   window.location.hash = where
}
</SCRIPT>
</HEAD>
```

```
<BODY>
<A NAME="start"><H1>Top</H1></A>
<FORM>
<INPUT TYPE="button" NAME="nest" VALUE="下一个" onClick="goNextAnchor('sec1')">
</FORM>
<HR>
<A NAME="sec1"><H1>选择第一部分</H1></A>
<FORM>
<INPUT TYPE="button" NAME="next" VALUE="下一个" onClick="goNextAnchor('sec2')">
</FORM>
<HR>
<A NAME="sec2"><H1>选择第二部分</H1></A>
<FORM>
<INPUT TYPE="button" NAME="next" VALUE="下一个" onClick="goNextAnchor('sec3')">
</FORM>
<HR>
<A NAME="sec3"><H1>选择第三部分</H1></A>
<FORM>
<INPUT TYPE="button" NAME="next" VALUE="返回到开始部分" onClick="goNextAnchor('start')">
</FORM>
<HR><P>
<SCRIPT LANGUAGE="JavaScript">
document.write("<I>在文档中总共定义了" + document.anchors.length + " 个 anchors 对象</I>")
</SCRIPT>
</BODY>
</HTML>
```

在浏览器中预览页面得到的效果如图 5-2 所示，当用户单击页面中的按钮时页面将发生改变。

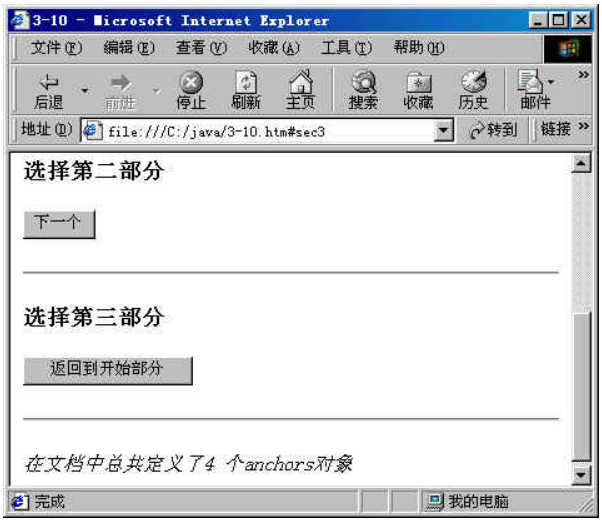


图5-32

5.15.3 Cookie 属　性

一段信息字符串值，它是由浏览器保存在客户端的 cookies.txt 文件中的。Cookie 是一个特殊的属性，它包含了客户机的状态信息，这些信息都可以被服务器访问。

1. 获　取 cookie

通过 JavaScript 程序获取的 cookie 数据存放在一个字符串中，包括整个名字——值对。下面我们将创建一个函数来获取 cookie 的值，下面就是函数的代码清单。

```
function GetCookie (name)
{
  var result = null;
  var myCookie = " " + document.cookie + ";";
  var searchName = " " + name + "=";
  var startOfCookie = myCookie.indexOf(searchName);
  var endOfCookie;
  if (startOfCookie != -1)
     {
          startOfCookie += searchName.length; // skip past cookie name
          endOfCookie = myCookie.indexOf(";", startOfCookie);
          result = unescape(myCookie.substring(startOfCookie, endOfCookie));
     }
  return result;
}
```

2. 设　置 cookie

可以利用 document.cookie 属性进行一个简单的 JavaScript 赋值操作，来将 cookie 数据写入到 cookie 文件中，但在写 cookie 值时一定要注意数据的格式是否正确，这是能够正确写入数据的关键所在。

下面创建的是一个设置 cookie 的函数，函数的代码清单如下。

```
function SetCookie (name, value, expires, path, domain, secure)
{
          var expString = ((expires == null) ? "" : ("; expires=" + expires.toGMTString()));
          var pathString = ((path == null) ? "" : ("; path=" + path));
          var domainString = ((domain == null) ? "" : ("; domain=" + domain));
          var secureString = ((secure == true) ? "; secure" : "");
          document.cookie = name + "=" + escape (value) +expString + pathString + domainString +
     secureString;
     }
```

5.15.4 Images 属　性

现在一种较为流行的用户界面技巧是，在用户把鼠标移动到代表可以单击的按钮的图像时，可以改变图像的外观；或者是用户在按下鼠标和松开鼠标按钮时更新图像。下面就利用 document.images 属性来实现这种方式。

下面的实例就是当鼠标在图像上滑动、按下、释放或移出时，改变相应的按钮图像。下面是实例的代码清单。

```
<HTML>
<HEAD>
<TITLE>图像的交换</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function setImage() {}
</SCRIPT>
<SCRIPT LANGUAGE="JavaScript">
if (navigator.appVersion.charAt(0) > 2)
{
   var RightNormImg = new Image(16,16)
   var RightUpImg = new Image(16,16)
   var RightDownImg = new Image(16,16)
   var LeftNormImg = new Image(16,16)
   var LeftUpImg = new Image(16,16)
   var LeftDownImg = new Image(16,16)
   RightNormImg.src = "RightNorm.gif"
   RightUpImg.src = "RightUp.gif"
   RightDownImg.src = "RightDown.gif"
   LeftNormImg.src = "LeftNorm.gif"
   LeftUpImg.src = "LeftUp.gif"
   LeftDownImg.src = "LeftDown.gif"
}
function setImage(imgName, type)
 {
   var imgFile = eval(imgName + type + "Img.src")
   document.images[imgName].src = imgFile
   return false
}
</SCRIPT>
</HEAD>
<BODY>
<B>请在图像按钮上操作鼠标:</B><P>
<CENTER>
<A HREF="javascript:void(0)"
   onMouseOver="return setImage('Left','Up')"
   onMouseDown="return setImage('Left','Down')"
   onMouseUp="return setImage('Left','Up')"
   onMouseOut="return setImage('Left','Norm')">
<IMG NAME="Left" SRC="LeftNorm.gif" HEIGHT=16 WIDTH=16 BORDER=0></A>
```

```
  
<A HREF="javascript:void(0)"
   onMouseOver="return setImage('Right','Up')"
   onMouseDown="return setImage('Right','Down')"
   onMouseUp="return setImage('Right','Up')"
   onMouseOut="return setImage('Right','Norm')">
<IMG NAME="Right" SRC="RightNorm.gif" HEIGHT=16 WIDTH=16 BORDER= 0></A>
</CENTER>
</BODY>
</HTML>
```

在上面的实例中我们同时使用了鼠标的事件，就是在鼠标的各个事件中我们调用了不同的图像来显示图片，这些图片都是在浏览器加载网页时预加载到浏览器缓存中的，这样当图像被调用时就可以立即得到响应。

5.15.5 forms 属　性

我们使用 document.forms 属性来定位表单对象或者元素。可以使用 document.forms[i] 来定位一个具体的表单元素。例如：为获取文档中的第三个表单的名称为“testphone”的输入文本域中的值，可以使用如下的语句来完成：

myphone=document.forms[2].testphone.value

在 document.forms 属性中还有一个比较重要的特性。JavaScript 中所有表示嵌入对象的数组都有一个 length 属性来返回文档中表单的元素个数。因此可以使用 document.forms.length 属性的返回值。

下面的实例是在文档中通过检查“blush”复选框的状态，将显示一个重复导向一个特殊音乐站点的警告对话框。在这里用户的输入分成两个表：一个表单是复选框，另外一个表单是导航按钮。下面就是实例的代码清单。

```
<HTML>
<HEAD>
<TITLE>document.forms 运用实例</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function goMusic() {
   if (document.forms[0].bluish.checked) {
      alert("现在转到联谊无限...")
   } else {
      alert("现在转到世纪星空...")
   }
}
</SCRIPT>
</HEAD>

<BODY>
<FORM NAME="theBlues">
<INPUT TYPE="checkbox" NAME="bluish">单击这里转到联谊无限.
</FORM>
```

```
<HR>
欢迎光临世纪联谊，精彩无限。
<BR>
<HR>
<FORM NAME="visit">
<INPUT TYPE="button" VALUE="音乐无限" onClick="goMusic()">
</FORM>
</BODY>
</HTML>
```

下面的代码就是利用 document.forms 来确定 bluish 单选按钮是否被选择，如果选择就弹出一个“正在转到联谊无限”对话框，否则弹出另外一个“正在转到世纪星空”对话框。

```
<SCRIPT LANGUAGE="JavaScript">
function goMusic() {
   if (document.forms[0].bluish.checked) {
      alert("现在转到联谊无限...")
   } else {
      alert("现在转到世纪星空...")
   }
}
</SCRIPT>
```

在浏览器中预览的页面效果如图 5-3 所示。

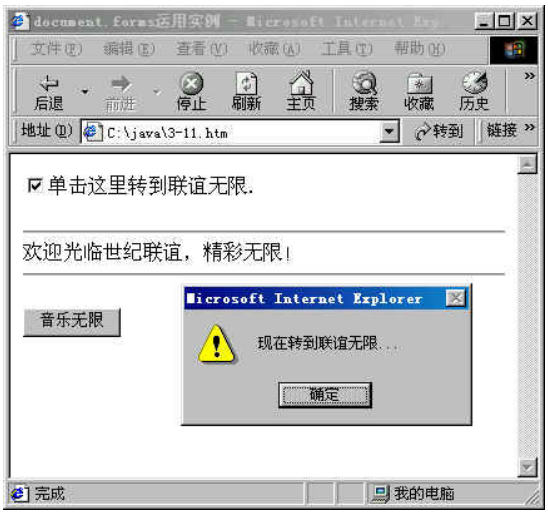


图5-33

5.16 document 对象的方法

5.16.1 Write（）和 Writeln（）方法

这两种方法都是将文本送到其窗口内的文档中显示。这两种方法的惟一区别在于 document.writeln（）方法在传送到文档中的字符串末时附加一个回车符。

在下面的例子中我们将利用 document.write（）方法创建一个 Web 页面，它利用系统时间的改变来修改页面的内容，在这个例子中我们将创建一个课程表，按照上午、下午和晚上分为三张，根据系统时间的变换，浏览器将自动改变 Web 页面中的内容。当检测到当前的系统时间在 0:00—12:00 之间，浏览器将显示上午的课程表；在 12:00—18:00 之间，浏览器将显示下午的课程表，在 18:00—24:00 之间，浏览器将显示晚上的课程表。

在创建该实例时，我们利用 date 对象的一些方法：getDate，getDay，getHours，getMinutes，getMonth，getSeconds，getTime，getyear 等。下面是实例的代码清单。

```
<HTML>
<BODY>
<SCRIPT LANGUAGE = JavaScript>
var currentDate = new Date()
var currentHour = currentDate.getHours()
document.write( "<CENTER>")
document.write( "<H1>")
document.write( "请详细查看今日的课程表")
document.write( "</H1>")
document.write( "<H2>")
document.write( "当前的日期和时间:")
document.write( currentDate.toLocaleString() )
document.write( "</H2>")
document.write( "</CENTER>")
if (currentHour < 6 || currentHour > 23){
        document.write( "<CENTER>")
        document.write( "<H1>")
        document.write( "休息" )
        document.write( "</H1>")
        document.write( "</CENTER>")
}
if (currentHour > 6 && currentHour < 12 )
        document.write( "<CENTER>")
        document.write( "<TABLE BORDER BGCOLOR = '#ffff00'>")
        document.write("<TR><TH COLSPAN = 2>上午课程安排</TH></TR>")
        document.write(<TR><TD>大学英语</TD>")
    document.write("<TD>8:00 AM</TD>")
    document.write("<TD>2A-3050 室</TD></TR>")
```

```
        document.write(<TR><TD>高等数学</TD>")
        document.write("<TD>9:00 AM</TD>")
        document.write("<TD>3B-3010 室</TD></TR>")
        document.write(<TR><TD>计算机原理</TD>")
        document.write("<TD>10:00 AM</TD>")
        document.write("<TD>2A-4008 室</TD></TR>")
        document.write(<TR><TD>大学物理</TD>")
        document.write("<TD>11:00 AM</TD>")
        document.write("<TD>2B-3020 室</TD></TR>")
            document.write("</TABLE>")
            document.write( "</CENTER>")
            document.write( "</TABLE>")
            document.write( "</CENTER>")
}
if ( currentHour >= 12 && currentHour < 18 )
            document.write( "<CENTER>")
            document.write( "<TABLE BORDER BGCOLOR = '#ffff00'>")
            document.write("<TR><TH COLSPAN = 2>下午课程安排</TH></TR>")
            document.write(<TR><TD>大学体育</TD>")
        document.write("<TD>2:30 PM</TD>")
        document.write("<TD>新体育馆</TD></TR>")
            document.write( "</TABLE>")
            document.write( "</CENTER>")
}
if ( currentHour >= 18 && currentHour < 22 )
            document.write( "<CENTER>")
            document.write( "<TABLE BORDER BGCOLOR = '#ffff00'>")
            document.write("<TR><TH COLSPAN = 2>晚上课程安排</TH></TR>")
            document.write(<TR><TD>计算机实习</TD>")
        document.write("<TD>20:00 AM</TD>")
        document.write("<TD>计算中心 4 室</TD></TR>")
            document.write( "</TABLE>")
            document.write( "</CENTER>")
}
</SCRIPT>
</HTML>
```

在上面的实例中利用 document.write()方法来创建 Web 页面的 HTML 语句。

而下面代码是利用 date 对象的 toLocalSting()方法来显示当前的系统日期和时间。

```
<SCRIPT LANGUAGE = JavaScript>
var currentDate = new Date()
var currentHour = currentDate.getHours()
//利用 date 对象的 getHours()方法来取得当前的系统日期。
document.write( "<CENTER>")
document.write( "<H1>")
document.write( "请详细查看今日的课程表")
document.write( "</H1>")
document.write( "<H2>")
document.write( "当前的日期和时间:")
```

```
document.write( currentDate.toLocaleString() )
```

//利用 date 对象的 toLocalSting()方法将系统日期时间转换为字符串，并且在浏览器中显示

```
document.write( "</H2>")
document.write( "</CENTER>")
```

下面的代码判断当前的系统时间，如果系统的时间在晚间 23:00 以后到早晨 6:00 之间，那么显示的是在夜间休息。

```
if (currentHour < 6 || currentHour > 23){
        document.write( "<CENTER>")
        document.write( "<H1>")
        document.write( "夜间休息" )
        document.write( "</H1>")
        document.write( "</CENTER>")
}
```

使用同样的方法判断当前的系统时间如果是在 8:00—12:00 之间，则显示上午的课程表，在表格中显示了安排的课程名称、上课的时间和上课的教室。下面就是完成该功能的代码：

```
if (currentHour > 6 && currentHour < 12 )
        document.write( "<CENTER>")
        document.write( "<TABLE BORDER BGCOLOR = '#ffff00'>")
    //使用表格显示上午的课程安排
        document.write("<TR><TH COLSPAN = 2>上午课程安排</TH></TR>")
        document.write(<TR><TD>大学英语</TD>")
    document.write("<TD>8:00 AM</TD>")
    document.write("<TD>2A-3050 室</TD></TR>")
    document.write(<TR><TD>高等数学</TD>")
    document.write("<TD>9:00 AM</TD>")
    document.write("<TD>3B-3010 室</TD></TR>")
    document.write(<TR><TD>计算机原理</TD>")
    document.write("<TD>10:00 AM</TD>")
    document.write("<TD>2A-4008 室</TD></TR>")
    document.write(<TR><TD>大学物理</TD>")
    document.write("<TD>11:00 AM</TD>")
    document.write("<TD>2B-3020 室</TD></TR>")
        document.write("</TABLE>")
        document.write( "</CENTER>")
        document.write( "</TABLE>")
        document.write( "</CENTER>")
}
```

同样判断当前的系统时间如果是在 12:00—18:00 之间，则显示下午的课程安排。

```
if ( currentHour >= 12 && currentHour < 18 )
        document.write( "<CENTER>")
        document.write( "<TABLE BORDER BGCOLOR = '#ffff00'>")
        document.write("<TR><TH COLSPAN = 2>下午课程安排</TH></TR>")
        document.write(<TR><TD>大学体育</TD>")
    document.write("<TD>2:30 PM</TD>")
```

```
        document.write("<TD>新体育馆</TD></TR>")
            document.write( "</TABLE>")
            document.write( "</CENTER>")
}
```

同样判断当前的系统时间如果是在 18:00—22:00 之间，则显示晚上的课程安排。

```
if ( currentHour >= 18 && currentHour < 22 )
            document.write( "<CENTER>")
            document.write( "<TABLE BORDER BGCOLOR = '#ffff00'>")
            document.write("<TR><TH COLSPAN = 2>晚上课程安排</TH></TR>")
            document.write(<TR><TD>计算机实习</TD>")
        document.write("<TD>20:00 AM</TD>")
        document.write("<TD>计算中心 4 室</TD></TR>")
            document.write( "</TABLE>")
            document.write( "</CENTER>")
}
```

最后在浏览器中浏览的效果如图 5-4 所示。

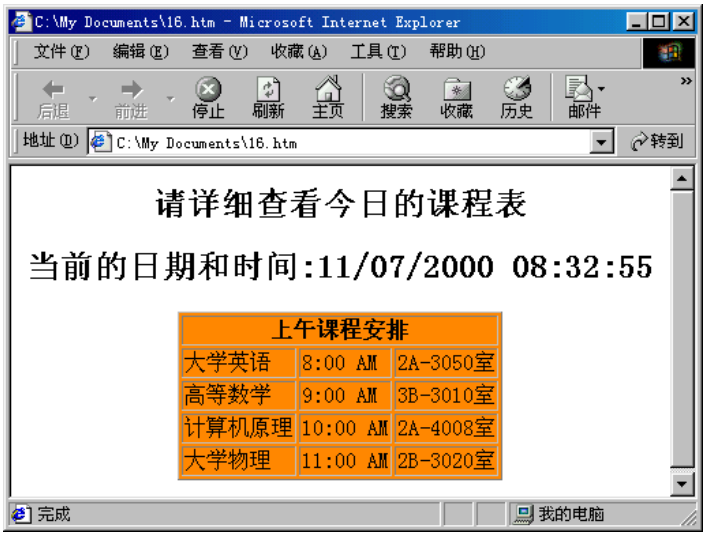


图5-34

5.16.2 Close（）方　法

无论任何时候用 document.open（）方法或任何一种文档的写方法为窗口打开了输出流以后，一旦写入文档就必须关闭输出流。这样对下一轮用新的 Script 装配 HTML 文档做好准备，所以关闭步骤是很重要的过程。如果不关闭窗口，后续的写文档就会自动追加到窗口的底部，直到使用 document.close（）方法，部分或者所有为窗口指定的数据才能正确显示，尤其是当图像作为文档流的一部分画出来时。在没有使用 document.close（）方法时文档的一部分瞬间出现又消失。如果以后出现这种情况，查看一下在最后的一个 document.write（）方法后是否缺少一个 document.close（）方法。

5.16.3 GetSelection（）方　法

可能很多用户都选择浏览器文档中文本，并将它粘贴到其它应用程序文档中。其实 Script 提供了捕获用户浏览页面中的文本。Document.getSelection（）方法会以字符串的形式返回用户选中的文本。如果没有选中任何东西，则返回的结果为空字符串。当然在使用 document.Selection（）方法返回的值只包含页面上可见的文本，并不包含底层 HTML 或者文本风格。

下面的实例使用了文档事件捕获，并且 getSelection（）方法在页面上将选中的文本内容显示在一个文本域对象中。下面是实例程序的代码清单：

```
<HTML>
<HEAD>
<TITLE>
document.getSelection 方法
</TITLE>
<SCRIPT LANGUAGE=”JavaScript1.2”>
function showSelection()
{
document.forms[0].selectedText.value=document.getSelection()
}
document.captureEvents(Event.MOUSEUP)
document.onmouseup=showSelection
</SCRIPT>
</HEAD>
<BODY>
<B>请选中页面中的一些文本</B>
<HR>
<P>
这些都是用来测试的文本，您可以用鼠标来选中，然后试一试选择文本后产生的结果。
</p>
<FORM>
<TEXTAREA NAME=”selectedText” ROWS=3 COLS=40 WRAP=”virtual”>
</textarer>
</FORM>
</BODY>
</HTML>
```

下面的代码是使用 JavaScript 调用 document 对象的 getSelection 方法来将选中的文本在文本区域内显示。

```
<SCRIPT LANGUAGE=”JavaScript”>
function showSelection()
{
document.forms[0].selectedText.value=document.getSelection()
}
document.captureEvents(Event.MOUSEUP)
document.onmouseup=showSelection
</SCRIPT>
```

实例程序在浏览器中预览的效果如图 5-5 所示。

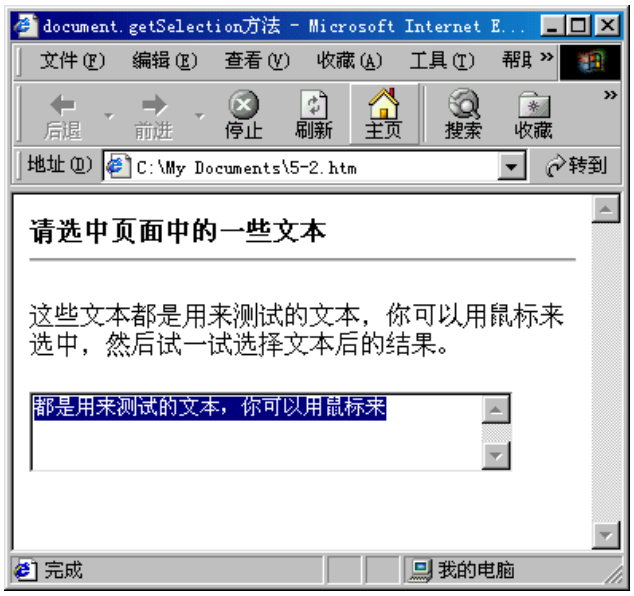


图5-35

第6章 文本对象

主要内容

- 文本对象的属性
- 文本对象的方法
- 文本对象的事件
- 文本区域对象

本章导读

文本对象是接收用户输入文本信息的主要方式。若在文本对象中显示用户的输入，可以将用户输入的文本信息提交给服务器处理。一般来说使用单行文本框作为用户输入，可以通过预测用户可能输入的字符数来设置文本对象框的大小，并且都以左对齐方式默认在文本对象中显示用户提交的信息。当然，如果用户需要提交多行文本，那么应该使用多行文本区域对象。

6.17 文本对象属性

在文本对象中主要定义如下的属性：defaultValue，form，name，type，value。在下面我们将通过实例来介绍文本对象的一些主要属性。

6.17.1 DefaultValue 属 性

对每个文本对象，用户都可以任意赋予一个满足要求的值。如果您在设计 Web 页面时，可以给输入文本对象赋予一个默认的字符串，这样浏览器总可以读取在<INPUT>定义赋予输入文本对象的字符串。DefaultValue 返回 VALUE=属性中的字符串参数。

在下面的实例中我们将创建一个简单的表单，它包含两个域，在第一个域中已经标记有一个缺省的值，而在另外一个域中没有赋予缺省的值。在页面中设置了一个按钮，当用户单击按钮时就将第一个域的值设置为缺省的值。程序的清单如下：

```
<HTML>
<HEAD>
<TITLE>
文本对象的 defaultValue 属性
</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function upperchar(field)
{
field.value=field.value.toUpperCase()
}
function resetfield(form)
{
form.conn.value=form.conn.defaultValue
}
</SCRIPT>
</HEAD>
<CENTER>
<FORM>
请输入英文字符：
<INPUT TYPE="text" NAME="conn" VALUE="aabbcc" onClick="upperchar(this)">
<BR>
<br>
<INPUT TYPE="button" NAME="mmnn" VALUE="恢复"onClick="resetfield(this.form)">
</FORM>
</CENTER>
</BODY>
</HTML>
```

下面的代码将在文本框中输入的英文字符串转换为大写形式。

```
function upperchar(field)
{
field.value=field.value.toUpperCase()
}
```

下面的代码将在文本框中输入的英文字符串清空，并且将设计程序时的缺省值赋予文本框。

```
function resetfield(form)
{
form.conn.value=form.conn.defaultValue
}
```

该实例在浏览器中预览得到的效果如图 6-1 所示。

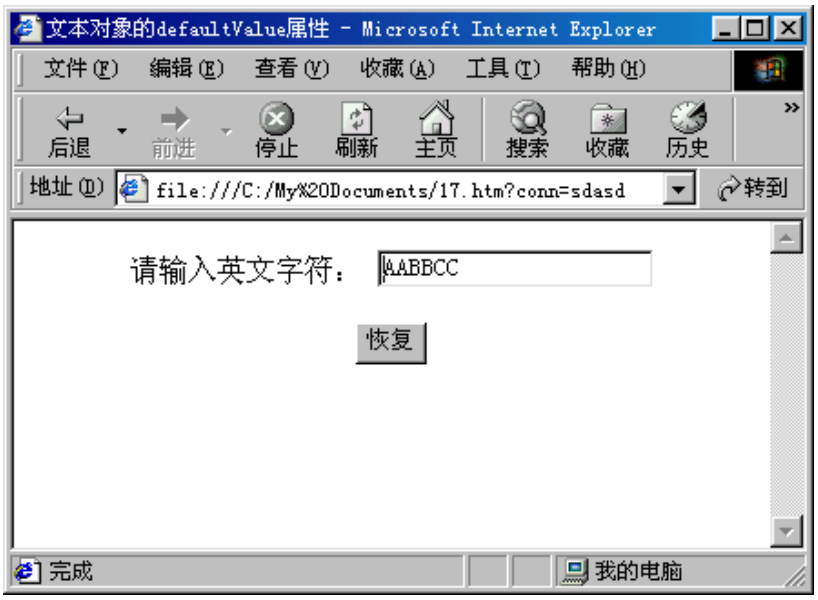


图6-36

6.17.2 Name 属　性

文本对象的名字是非常重要的一个属性，它用来区分在 Web 页面中的多个对象。特别是在 HTML 页面向服务器提交数据信息时，输入的文本对象名字和数据都一同提交到服务器程序，这样来区分从表单提交的数据信息，同时也是其它程序引用该文本对象的关键所在。所以我们在设计 Web 页面时都要为每个文本对象赋予一个有意义的名字，以区别其它的文本对象。

关于 name 属性的运用已经在其它实例中多次使用，在这里就不再举例。

6.17.3 value 属　性

文本对象的 value 属性是用来读取和设置域中的信息。文本对象赋予的值都是字符串信息，不论赋予的值是字符串、数字还是布尔变量，它们都会自动转换为字符串。因此我

们在取回文本对象域中的值时得到的都是字符串形式。

关于 value 属性的运用已经在其它实例中多次使用，在这里就不再举例。

6.18 文本对象的方法

文本对象主要有以下的方法：blur（），focus（），handleEvent（），select（）等。在下面我们将主要介绍文本对象的 focus（）和 select（）两种方法。

6.18.1 Focus（）方 法

聚焦就是在多个文本对象中把光标放置在某个对象的域中指定的位置。一般来说是将光标放置在文本对象域的文本的开头处。

在下面的实例中我们将详细介绍关于 focus()方法的运用。实例的代码清单如下：

```
<html>
<head>
<title>Online Registration</title>
</head>
<body>
<form name="form1" method="POST">
<h2>Online Registration</h2>
<p>Username:<br>
<input type=text size=25 maxlength=256 name="Username"
onFocus="this.select()"><br>
Browser used:<br>
<input type=text size=25 maxlength=256 name="Browser"
onFocus="this.select()"><br>
Email address:<strong> <br>
</strong><input type=text size=25 maxlength=256 name="EmailAddress"
onFocus="this.select()"></p>
<h2><input type=submit value="Register"> <input type=reset value="Clear"></h2>
</form>
</body>
</html>
```

实例程序在浏览器中预览的效果如图 6-2 所示。

图6-37

6.18.2 Select（）方　法

在一个指定的域中选定文本对象中的文本。当文本对象一经选定，用户就可以使用 del 键删除以前输入文本框中的信息。我们选定一个文本对象不会触发相同的 onSelect=事件处理程序，除非是我们自己手动选定另外一个事件处理程序。

在下面的实例中将运用文本对象的 Select（）方法。实例是通过用户单击检验按钮发送文本对象的内容来进行数字验证。如果用户输入的不是数字，则将为用户预选输入域。在程序中对文档的焦点定位。下面是程序的清单：

```
<HTML>
<HEAD>
<TITLE>
文本对象的 focus()方法
</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function isNum(inputStr)
{
for(var I=0; I<inputStr.length; I++)
{
var oneChar=inputStr.charAt(I)
if (oneChar<"0"|| oneChar>"9")
{
alert("请你只输入数字")
return false
}
}
```

```
return true
}
function checklt(form)
{
inputStr=form.numeric.value
if (isNum(inputStr))
{
}
else
{
form.numeric.focus()
form.numeric.select()
}
}
</SCRIPT>
</HEAD>
<BODY>
<CENTER>
<FORM>
请输入数字：
<INPUT TYPE="text" NAME="numeric">
<BR>
<INPUT TYPE="button" VALUE="检验" onClick="checklt(this.form)">
</FORM>
</CNETER>
</BODY>
</HTML>
```

下面的代码检验输入的字符串是否是数字，如果不是数字，则弹出对话窗口，要求用户输入数字字符串。

```
function isnum(inputStr)
{
for (var I=0;I< inputStr.length;I++)
{
car oneChar=inputStr.charAt(I)
if oneChar<”0”|| oneChar>”9”)
{
alert(“请您只输入数字”)
return false
}
}
return true
}
```

下面的代码是响应检验数字字符串的函数，当检验到不是数字字符串时，返回输入窗口，并且将焦点定位在文本框中。

```
function checklt(form)
{
inputStr=form.numeric.value
```

```
if (isNum(inputStr))
{
}
else
{
form.numeric.focus()
form.numeric.select()
}
}
```

6.19 文本对象的事件

在所有关于文本对象的事件处理程序中，在表单中可能最常用的是 onChange（）事件。对于用户在该域内输入的任何内容的校验，该事件将起到重要的作用。在提交表单之前，对所有输入仅进行批处理模式校验是潜在危险的，因为此时用户已经离开了一个给定区域的输入数据。

在下面的实例中，无论用户何时改变文本，然后点击离开按钮，onChange（）事件都将被送到该区域中，并启动 onChange（）事件处理程序。下面是实例代码的程序清单：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD><TITLE>文本对象事件</TITLE>
<META content="text/html; charset=gb2312" http-equiv=Content-Type>
<SCRIPT language=JavaScript>
function isNumber(inputStr)
{
for (var I=0;I<inputStr.length;I++)
{
var oneChar=inputStr.substring(I,I+1)
if (oneChar<"0" || oneChar>"9")
{
alert("只能输入数字字符")
return false
}
}
return true
}
function checkit(form)
{
inputStr=form.numeric.value
if (isNumber(inputStr)==true)
{
}
```

```
else
{
form.numeric.focus()
form.numeric.select()
}
}
</SCRIPT>

<META content="MSHTML 5.00.2614.3500" name=GENERATOR></HEAD>
<BODY>
<FORM>请输入数字字符：<INPUT name=numeric onchange=checkit(this.form)>
<P></FORM></P></BODY></HTML>
```

6.20 文本区域对象

在表单中我们可以使用不同的对象来提交信息，一般的对象所能提交的信息都不能超过 256 个字符，如果输入的元素需要超过 256 个字符，那么我们只能使用文本区域对象。因为文本区域对象与文本对象非常相似，但是文本区域对象可以使用多行的文本输入，所以我们在表单中设计文本区域对象时，必须指明文本区域对象的属性包括高度（行数）和宽度（列数）。

文本对象的所有属性、方法和事件都可以应用于文本区域对象。它们的使用方法基本相同（当然“textarea”的 type 属性除外）。

在下面的实例中，我们将创建一个包含一个文本区域对象和按钮的 Web 页面。在实例中我们自己定义了文本区域对象的行数和列数，用户可以在文本区域中输入提交信息，如果用户输入的信息超过了文本区域的行数，则文本区域对象会自动增加滚动条。程序清单如下：

```
<HTML>
<TITLE>
文本区域对象实例
</TITLE>
<BODY>
<CENTER>
<FORM>
<TEXTAREA NAME = "Textarea" COLS = 20    ROWS = 15>
这是一个文本区域对象的实例
</TEXTAREA>
<BR>
<BR>
<INPUT TYPE = BUTTON Value = "显示信息" onClick = "DisplayMessage(this.form)">
</FORM>
</CENTER>
```

```
</BODY>
<SCRIPT LANGUAGE = JavaScript>
    function DisplayMessage(form1)
    {
        form1.Textarea.value = "Hello from JavaScript"
    }
</SCRIPT>
</HTML>
```

下面的代码是用来清除文本区域对象中的文本，并且赋予另外一文本信息。

```
<SCRIPT LANGUAGE = JavaScript>
    function DisplayMessage(form1)
    {
        form1.Textarea.value = "Hello from JavaScript"
    }
</SCRIPT>
```

实例在浏览器中预览效果如图 6-3 所示。

图6-38

第7章 按钮对象

主要内容

- 按钮对象
- 复选框对象
- Radio 对象

本章导读

本章主要通过具体的实例来介绍在 Web 页面中常用的三种按钮对象：普通按钮、提交按钮、复位按钮对象；以及复选框对象和 Radio 对象在 Web 页面中的运用。

7.21 button、submit、reset 对象

在 HTML 表单中定义了三种类型的按钮：按钮（button）、提交（submit）和复位（reset）对象。使用 HTML 语法定义按钮如下：

```
<INPUT
        TYPE=”button|submit|reset”
        NAME=”objectname”
        VALUE=”labeltext”
        OnClick=”methodName”>
```

三种类型的按钮看起来非常相似，但是它们的使用目的却不一样，它们的使用目的定义如下：

- **提交（submit）按钮**：提交按钮对象是用来提交数据信息给服务器的处理程序，JavaScript 代码不干涉该按钮的活动，且这些都是按钮本身属性完成的;
- **复位（reset）按钮**：复位按钮对象是用来清除表单对象中各个文本域的文本信息，并且将各个缺省的值赋予文本对象域。同理，JavaScript 代码也是不干涉该按钮的活动，且这些都是按钮本身属性完成的;
- **按钮（button）**：这是一个普通的按钮对象，为了使用该按钮完成某项工作，需要增加一个 onClick 处理事件来驱动该按钮。

在下面的实例中我们将使用三种类型按钮。使用提交按钮来提交信息，使用复位按钮来清除表单中的文本域中信息，使用按钮来显示一个帮助用户填充注册信息的窗口。下面是实例程序的代码清单：

```
<html>
<head>
<title>Online Registration</title>
<SCRIPT LANGUAGE="JavaScript">
    function showHelp() {
        helpWin = window.open("", "Help", "height=200,width=400")
        helpWin.document.write("<body><h2>Help on Registration</h2>")
        helpWin.document.write("1. Please enter your product information into the fields.<p>")
        helpWin.document.write("2. Press the Register button  to submit your form.<p>")
        helpWin.document.write("3. Press the Clear button to  clear the form and start again.<p>")
        helpWin.document.write("<p>")
        helpWin.document.write("<form><DIV ALIGN='CENTER'>")
        helpWin.document.write("<input type=button value=' OK 'onClick ='window.close()'>")
        helpWin.document.write("</DIV></form></body>")
    }
</SCRIPT>
```

```
</head>
<body>
<h1>Online Registration</h1>
<form method="POST">
<p>Please provide the following product information:</p>
<blockquote>
<pre><em>       Product name </em><input type=text size=25 maxlength=256
name="ProductName">
<em>                  Model </em><input type=text size=25 maxlength=256
name="Product_Model">
<em>    Version number </em><input type=text size=25 maxlength=256
name="Product_VersionNumber">
<em>Operating system </em><input type=text size=25 maxlength=256
name="Product_OperatingSystem">
<em>     Serial number </em><input type=text size=25 maxlength=256
name="Product_SerialNumber">
</pre>
</blockquote>
<p><input type=submit value="Register"> <input type=reset value="Clear">
<input type=button value="Help" onClick="showHelp()"></p>
</form>
</body>
```

下面的代码是在用户单击帮助按钮后产生的 Web 页面。

```
<SCRIPT LANGUAGE="JavaScript">
 function showHelp() {
        helpWin = window.open("", "Help", "height=200,width=400")
        helpWin.document.write("<body><h2>Help on Registration</h2>")
        helpWin.document.write("1. Please enter your product information into the fields.<br>")
        helpWin.document.write("2. Press the Register button    to submit your form.<br>")
        helpWin.document.write("3.  Press  the  Clear  button  to      clear  the  form  and  start
again.<br>")
        helpWin.document.write("<p>")
        helpWin.document.write("<form><DIV ALIGN='CENTER'>")
        helpWin.document.write("<input type=button value='    OK    'onClick='window.close()'>")
        helpWin.document.write("</DIV></form></body>")
 }
</SCRIPT>
```

实例程序在浏览器中预览的效果如图 7-1 所示。

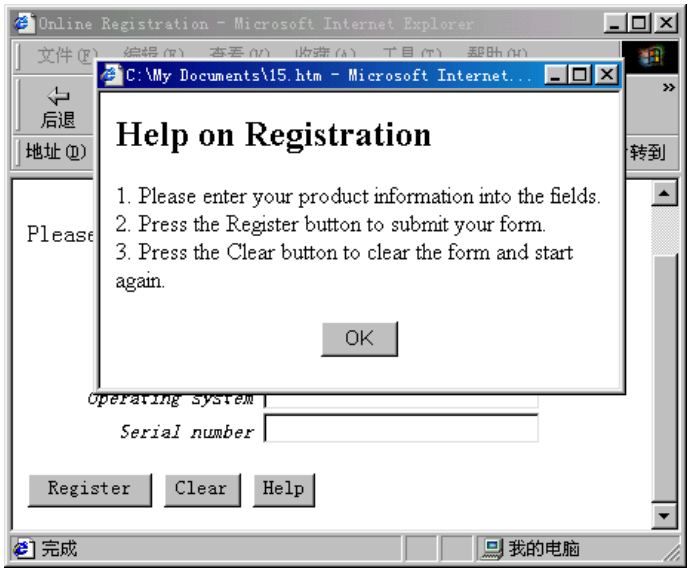


图7-39

7.22 复选框对象

复选框是图像界面下一个特殊的标识，它是表单上的标志，指出的数据信息是真或者是假。当某个复选框没有被标记或者为空，则这个数据信息标记就是假或者不被包括。如果两个以上的复选框被聚成组，它们一般是互不影响，每个复选框都是一个独立的设置（这与我们后面介绍的 radio 对象是有区别的）。

使用 HTML 定义一个复选框的语法如下：

```
<INPUT
    TYPE=”checkbox”
    NAME=”objectName”
    VALUE=”value”
    CHECKED
    OnClick=”methodName”>
```

复选框主要具有以下的属性：checked、defaultChecked、name、Value 等。下面我们将介绍与其它对象不同的属性。

Checked 属性是复选框最简单、最有用的属性，它给出用户设置或者是我们在设计程序时赋予复选框的设置，一个复选框是被选还是没有被选，对于被选的其值是真，对于没有被选的其值是假的。如果要由 JavaScript 语言来选中一个复选框，只要赋予该复选框的 checked 属性一个真值即可。例如：

Document.forms[0].boxName.checked=true

用这样一条 JavaScript 语句来设置复选框的 checked 属性，实质上是触发了复选框的单击事件。

在下面我们将举一个实例来讲述怎样利用复选框的 checked 属性，并且可以将复选框的 checked 的属性值作为 if...else 条件语句来判断测试的结果。下面是实例的程序清单：

```
<HTML>
<HEAD>
<TITLE>
复选框的实例
</TITLE>
<SCRIPT LANGUAGE=”JavaScript”>
function checkBox(form)
{
if (form.checkThis.checked )
{
alert(“这个复选框已经被选择”)
}
else
{
alert(“这个复选框现在还没有被选择”)
}
}
</SCRIPT>
</HEAD>
<BODY>
<CENTER>
<FORM>
<INPUT TYPE=”checkbox” NAME=”checkThis”>
请单击按钮
<BR>
<INPUT TYPE=”button” NAME=”boxCheck” VALUE=” 检 测 按 钮 ” onClick
=”checkBox(this.form)”>
</FORM>
</CENTER>
</BODY>
</HTML>
```

在浏览器中浏览该实例，得到的效果如图 7-2 所示。

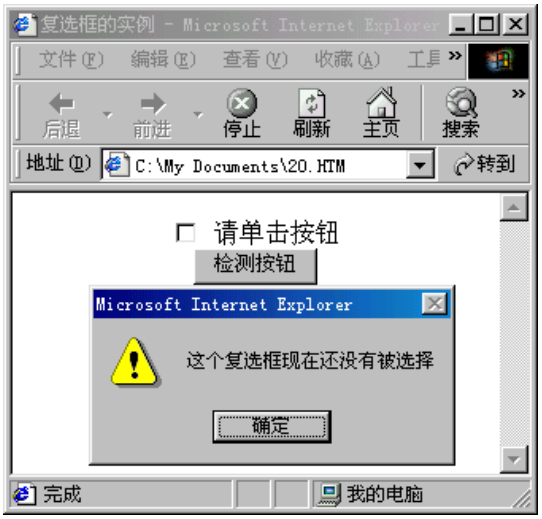


图7-40

7.23 Radio 对象

使用单选按钮对象是让用户在一组选项中选择一个单一的选项，假如选择一个选项，那么其它选项就不能再被选择。

使用 HTML 语法定义 Radio 对象的格式如下：

```
<INPUT
    TYPE=”radio”
    NAME=”groupName”
    VALUE=”value”
    CHECKED
    OnClick=”methodName”
>
```

单选按钮对象主要具有如下的属性：checked、defaultChecked、length、name、type、value 等，我们将在下面通过实例来介绍单选按钮对象的最常见和最有用的属性。

在实例中我们将在 Web 页面中创建三个单选按钮对象，并且创建一个普通按钮，当用户单击普通按钮时，将弹出一个对话窗口显示在 Web 页面中的 Radio 对象中的某个单选按钮被选择。下面是实例的代码清单：

```
<html>
<head>
<script language = "JavaScript">
    function getRadioValue(radioObject) {
        var value = null
        for (var i=0; i<radioObject.length; i++) {
            if (radioObject[i].checked) {
                value = radioObject[i].value
                break }
        }
        return value
    }
</script>
</head>
<body>
    <center>
<form name="form1">
<input type=radio name="songs" value="星期一">星期一
<br>
    <input type=radio name="songs" value="星期二">星期二
    <br>
```

```
        <input type=radio name="songs" value="星期三">星期三
<br>
        <input type=radio name="songs" value="星期四">星期四
<br>
        <input type=radio name="songs" value="星期五">星期五
<br>
        <input type=radio name="songs" value="星期六">星期六
<br>
        <input type=radio name="songs" value="星期日">星期日
<br>
        <input type=button value="请选择"
        onClick="alert(getRadioValue(this.form.songs))">
</form>
        </center>
</body>
</html>
```

下面的代码是利用 Script 语言来处理单选按钮对象的过程。

```
        <script language = "JavaScript">
         function getRadioValue(radioObject) {
                var value = null
                for (var i=0; i<radioObject.length; i++) {
                        if (radioObject[i].checked) {
                                value = radioObject[i].value
                                break }
                }
                return value
        }
</script>
```

在浏览器中预览的效果如图 7-3 所示。

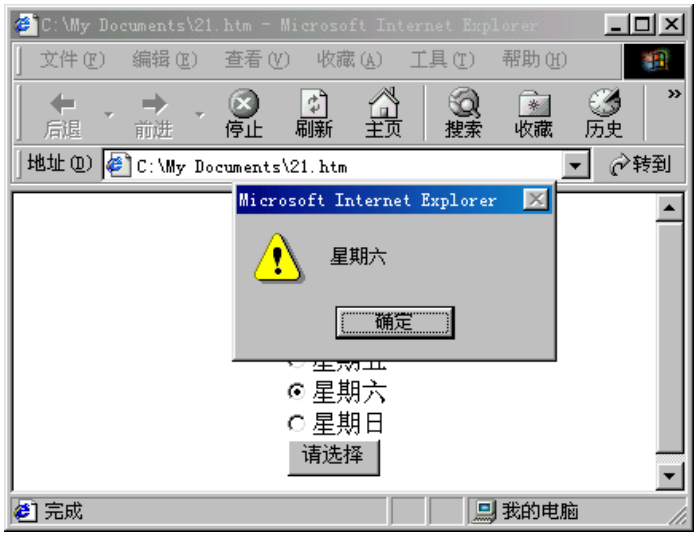


图7-41

第8章 选择和隐藏对象

主要内容

- 选 择 对 象
- 隐 藏 对 象

本章导读

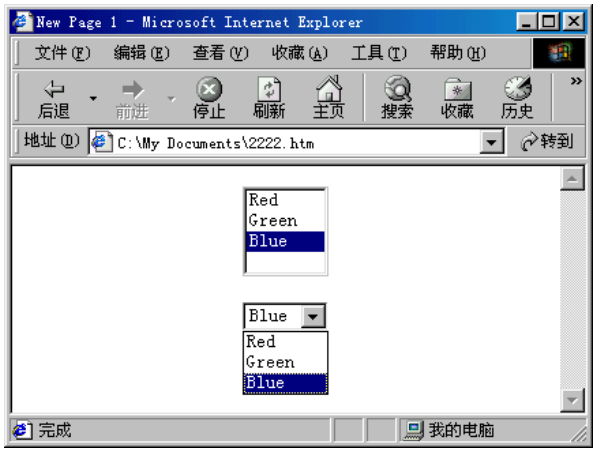
本章主要通过两个具体的实例介绍了选择对象和隐藏对象运用在 Web 中用户作出的选择结果处理方式。

8.24 select 对象

Select 对象是提交表单对象中最有用和最具有吸引力的对象之一。它们可以在一个表单中以弹出列表方式出现在 Web 页面中，也可以以滚动列表框的方式出现在 Web 页面中。其中弹出列表的方式是设计 Web 页面中最常见的方式，它只把用户选择的项显示在 Web 页面中，其它的选项则隐藏在列表框中。以免一些复杂的信息引起 Web 页面的混乱。

当定义一个表单内的 Select 对象时，使用的 HTML 语法如下：

```
<SELECT
        NAME=”objectName”
        SIZE=”numberVisible”
        MULTIPLE
        OnBlur=”methidName”
        OnChange=”methodName”
        OnFocus=”methodName”
        <OPTION VALUE=”optionValue”
        SELECTED>displayText</OPTION>
            <OPTION VALUE=”optionValue”
        SELECTED>displayText</OPTION>
</SELECT>
```



在定义 Select 对象时，其中 SIZE 属性确定了 Select 对象是作为弹出列表方式显示还是以列表方式显示。如果我们在设计时忘记设置这个属性，那么浏览器将自动地将 SIZE 的属性赋予 1 为缺省值。这样就使得 Select 对象是以弹出列表的方式出现在 Web 页面中的。如图 8-1 所示是我们在设置 SIZE 属性的值为 4 和为 1 时的效果。

在下面的实例中我们将利用 JavaScript 语言来选择 Select 对象中的选项。实现该功能是通过利用 select 对象的 options 数组特性。该实例程序的代码清单如下：

```
<html>
<head>
<script language = "JavaScript">
    function showSelection(objectName) {
        var list = ""
        for (var i=0; i<objectName.length; i++) {
            if (objectName.options[i].selected) {
                list += objectName.options[i].text + "<p>"
```

```
                }
            }
            selWindow = window.open("", "Selections", "height=200,width=400")
            selWindow.document.write("<h2>您选择的城市是：</h2><p><p>")
            selWindow.document.write(list)
        }
</script>
</head>
<body>
    <center>
    <form name="form1">
请在下面选择您喜欢的城市：<p>
    <select NAME="songs" SIZE=5 MULTIPLE>
    <option>北京</option>
    <option>上海</option>
<option>天津</option>
<option>重庆</option>
<option>成都</option>
<option>西安</option>
<option>拉萨</option>
<option>广州</option>
<option>深圳</option>
<option>长春</option>
</SELECT><p>
<input type=button value="显示"
onClick="showSelection(this.form.songs)">
</form>
    </center>
    </body>
</html>
```

下面的代码是利用 select 对象的 option 数组特性来显示用户所做出的选择。它通过数组的边界作为循环的变量上限，然后由 select 对象的 selected 属性的逻辑布尔值作为循环条件，进一步判断用户的选择。

```
    <script language = "JavaScript">
     function showSelection(objectName) {
            var list = ""
            for (var i=0; i<objectName.length; i++) {
                if (objectName.options[i].selected) {
                    list += objectName.options[i].text + "<p>"
                }
            }
            selWindow = window.open("", "Selections", "height=200,width=400")
            selWindow.document.write("<h2>您选择的城市是：</h2><p><p>")
            selWindow.document.write(list)
    }
</script>
```

代码程序在浏览器中预览的效果如图 8-2 所示。

图 8-2

8.25 隐含对象

隐含对象是表单对象中一个简单的字符串占用者，该对象对用户的 Web 页面是不可见的。隐含对象没有任何方法和事件处理程序，只是作为字符串的传送工具，起到一个中间变量的作用，恰好这些字符串是 Script 语言需要引用的值或者是其它链接的数据。

当定义一个表单内的隐含对象时，使用的 HTML 语法如下：

```
<INPUT
        TYPE=”hidden”
        NAME=”objectName”
        VALUE=”value”>
```

例如下面的一行就是定义一个隐含的对象代码：

```
<INPUT TYPE=”hidden” NAME=”hiddentext”>
```

在下面的实例中，我们首先在 Web 页面中创建了几个单选按钮对象，要求用户做出选择，如果用户做了多次选择，那么用户单击恢复按钮，可以将用户上次的选择恢复。这实际就是使用的隐含对象来作为中间变量，即以用户上次的选择结果，当需要恢复时就将保存在隐含对象中的值传递给所需的变量。实例程序的代码清单如下：

```
<html>
<head>
<title>Hidden Test</title>
<script language="JavaScript">
    function postData(value)
    {
        document.form1.holder2.value = document.form1.holder.value
        document.form1.holder.value = value
    }
```

```
        function resetValue()
        {
              var len = document.form1.ctyList.length
              for (var i=0; i<len; i++)
        {
                    if (document.form1.ctyList[i].value ==
                        document.form1.holder2.value)
       {
       document.form1.ctyList[i].checked = "1"
      break
       }
       }
        }
</script>
</head>
<body>
<h1>
选择您喜欢的国家
      </h1>
      <center>
<form name="form1" method="POST">
<br>
      <input type=radio name="ctyList" value="中国"
onClick="postData(this.value)">中国
<br>
      <input type=radio name="ctyList" value="美国"
onClick="postData(this.value)">美国
<br>
      <input type=radio name="ctyList" value="日本"
onClick="postData(this.value)">日本
<br>
      <input type=radio name="ctyList" value="德国"
onClick="postData(this.value)">德国
<br>
      <input type=radio name="ctyList" value="法国"
onClick="postData(this.value)">法国
<br>
      <input type=radio name="ctyList" value="瑞士"
onClick="postData(this.value)">瑞士
<br>
      <input type=radio name="ctyList" value="丹麦"
onClick="postData(this.value)">丹麦
<br>
      <input type=radio name="ctyList" value="韩国"
onClick="postData(this.value)">韩国
<br>
      <input type=button name="UndoLast" value="恢复上一次"
onClick="resetValue()">
```

```
<INPUT TYPE="hidden" NAME="holder" value="">
<INPUT TYPE="hidden" NAME="holder2" value="">
</form>
    </center>
</body>
</html>
```

下面的代码是提交用户的选择信息给隐含对象。

```
function postData(value)
 {
        document.form1.holder2.value = document.form1.holder.value
        document.form1.holder.value = value
 }
```

下面的代码是从隐含对象中恢复数据信息。

```
function resetValue()
 {
        var len = document.form1.ctyList.length
        for (var i=0; i<len; i++)
 {
                if (document.form1.ctyList[i].value ==
                    document.form1.holder2.value)
{
document.form1.ctyList[i].checked = "1"
break
}
}
                }
```

实例程序在浏览器中预览的效果如图 8-3 所示。

图 8-3

第9章 location 对 象

主要内容

- hash 属 性
- Href 属 性
- pathname 属 性
- Protocol 属 性

本章导读

location 对象提供关于任何当前打开的窗口或者特定的框架的 URL 信息。虽然在一个多框架窗口中，在 location 域中显示的是父窗口的 URL 信息，但是每个框架都有各自的 location 对象。它们在浏览器中是不可见的，但是我们在设计时是可以引用的。

location 对象的多数属性都是面向网络的处理信息，这些信息包括了网络上文档的物理位置的数据，还包含了主服务器、所使用的协议、端口等其它组件。location 对象的属性主要如下：protocol、hostname、port、pathname、hash、href 等。在下面我们将介绍 location 对象的一些主要的属性。

9.26 hash 属 性

hash 是一个 URL 习惯用法，它是以字符“#”开始指向一个位于文档中的 anchor，使浏览器打开一个新的 URL。一个 location 对象的 hash 属性是当前 URL 的 anchor 部分的名字（它由 hash 标记符和名字组成）。

如果在设计 Web 页面中已经带有 anchor 的 HTML 文档，并且链接到那些 anchor，此时目标位置将 anchor 显示为 URL 的一部分，当用户人工滚动到已经定义的 anchor 的文档位置时，那么窗口的 anchor 值将不再改变。

在下面的实例中我们将利用 location 对象的 hash 属性来调整浏览器中的 Web 页面的位置。下面是实例程序的代码清单：

```
<html>

<head>
<title> location.hash 属性 </title>
<script LANGUAGE="JavaScript">
function goNext(where)
{
window.location.hash=where
}
</script>
</head>
<a NAME="start">

<body>

<h2 align="center">顶端 </h2>
</a>

<form>
  <p><input TYPE="button" NAME="next" VALUE="NEXT" onClick="goNext('sec1')"> </p>
</form>

<hr align="center">
<a NAME="sec1">

<h2 align="center">第一部分 </h2>
</a>

<form>
  <p><input TYPE="button" NAME="next" VALUE="NEXT" onClick="goNext('sec2')"> </p>
```

```
</form>

<hr align="center">
<a NAME="sec2">

<h2 align="center">第二部分 </h2>
</a>

<form>
   <p><input TYPE="button" NAME="next" VALUE="NEXT" onClick="goNext('sec2')"> </p>
</form>

<hr align="center">
<a NAME="sec3">

<h2 align="center">第三部分 </h2>
</a>

<form>
   <p><input TYPE="button" NAME="next" VALUE="BACK TO TOP" onClick="goNext('start')"> </p>
</form>
</body>
</html>
```

下面的代码是其在响应按钮时，将文档中的 anchor 传递给浏览器，并指向一个新的 URL。

```
<SCRIPT LANGUAGE=”JavaScript”>
function goNext(where)
{
window.location.hash=where
}
</SCRIPT>
```

实例在浏览器中预览的效果如图 9-1 所示。

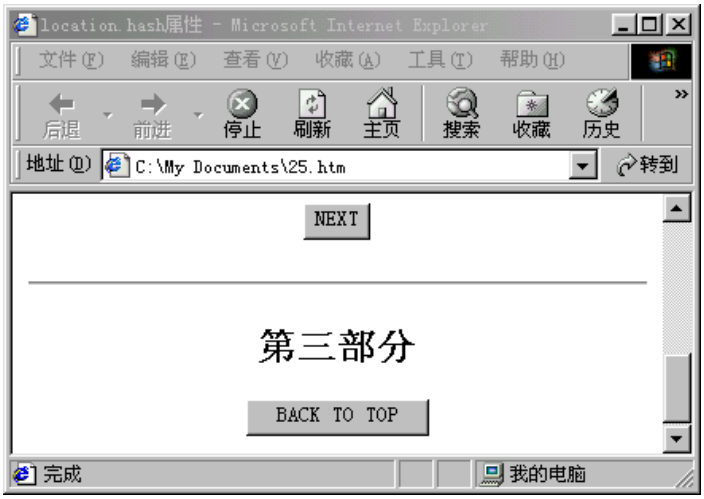


图9-42

9.27 Href 属 性

在所有 location 对象属性中，href（超文本引用）是在编写 Script 过程中最常被调用的一个属性。location.href 属性提供了一个指定窗口对象的整个 URL 的字符串。在赋值语句的左边使用该属性是在窗口中显示打开的 URL 的 JavaScript 方法。任何一条语句都可以在一个单框架浏览器窗口中加载 Web 站点的索引网页：

```
window.location=”http://www.yahoo.com”
window.location.href=”http://www.yahoo.com”
```

在不同的浏览器中，location.href 属性的值可能会被非字母和数字的 ASCII 码等来编码。这些 ASCII 编码中包括%符号和 ASCII 数字值。如果需要获取一个 URL，并且在创建的文档中将显示为一个字符串，最安全的方法是通过 JavaScript 内部的 unescape()函数来传递所有这种编码的字符串。

在下面的实例中，我们将使用 JavaScript 内部的 unescape()函数来处理捕获的 URL。然后在警告对话框中显示经过 ASCII 编码的路径名。实例的代码清单如下：

```
<html>

<head>
<title> 显示路径 </title>
<script LANGUAGE="JavaScript">
function getDirPath(URL)
{
var result = unescape(URL.substring(0,(URL.lastIndexOf("/"))+1))
return result
}
function showDir(URL)
{
alert(getDirPath(URL))
}
</script>
</head>

<body>

<form>
   <p><input TYPE="button" VALUE="查看 URL 的路径"
   onClick="showDir(window.location.href)"> </p>
</form>
</body>
</html>
```

下面的代码是用来取得 URL 的路径和显示 URL 的路径。

```
<Script LANGUAGE=”JavaScript”>
function getDir(URL)
{
var result=unescape(URL.substring(0,(URL.lastindexOf(“/”))+1))
```

```
return result
}
function showDir(URL)
{
alert(getDir(URL))
}
</SCRIPT>
```

9.28 pathname 属 性

location.pathname 属性的 URL 的路径名称部分是由服务器的 Root（根）相关的目录结构组成。也就是说根（在一个 http：连接的服务器名）不是路径名的一部分。如果 URL 的路径是通向根目录的一个文件，那么 location.pathname 属性是一个反斜杠（/）。任何其它路径名是以反斜杠（/）开始来指明一个嵌套在根内的目录。

在前面我们已经讲解了关于运用 location.pathname 属性的实例，在这里我们不再讲述。

9.29 Protocol 属 性

任何 URL 的第一个组成部分是用于特定通信类型的协议。下面的列表是目前使用最多的各种通信协议。

Web	http:
File	file:
FTP	ftp:
MailTo	mailto:
Usenet	news:
Gopher	gopher:
JavaScript	javascript:
Navigator	about:

对于最常用的 World Wide Web 网页来说，Hypertext Transfer Protocol（http）（超文本传输）协议是标准协议。在 location.protocol 属性中不仅包括协议名，还包括紧跟着的冒号分界符，这样对于一个典型的 Web 网页 URL 的 location.protocol 属性是：

http:

通常在 URL 中协议后的斜杠符不是 location.protocol 属性的一部分。在所有的 location 对象属性中，只有完整的 URL（location.href）显示协议和其它部分间的斜杠分界符。

在下面的实例中我们将运用 location.protocol 属性来创建一个 Web 页面，下面是实例

程序的代码清单：

```
<html>
<head>
<title>
location.protocol Example
</title>
<SCRIPT LANGUAGE="JavaScript">
<!--
        function gotoPage() {
          parent.frames[0].location.href = window.document.loc.ProtocolField.
            options[window.document.loc.ProtocolField.selectedIndex].text+
document.loc.HostnameField.value + document.loc.PathnameField.value
        }
//-->
</SCRIPT>
</head>
<body>
<p><font size=5>protocol//hostname:port pathname</font></p>
<form name="loc" method="POST">
<pre>Protocol:
<select name="ProtocolField" size=1>
<option>http://</option>
<option>file://</option>
<option>javascript:</option>
<option>ftp:</option>
<option>mailto:</option>
<option>gopher:</option>
<option>about:</option>
</select>
Hostname:
<input
type="text"
size=23
maxlength=256
name="HostnameField"
value="www.yahoo.com">
Pathname:
<input
type="text"
size=20
maxlength=100
name="PathnameField"
value="/">
<input
type="button"
name="Go"
value="Go"
onClick="gotoPage()">
```

```
</pre>
</form>
</body>
</html>
```

实例程序在浏览器中预览的效果如图 9-2 所示。

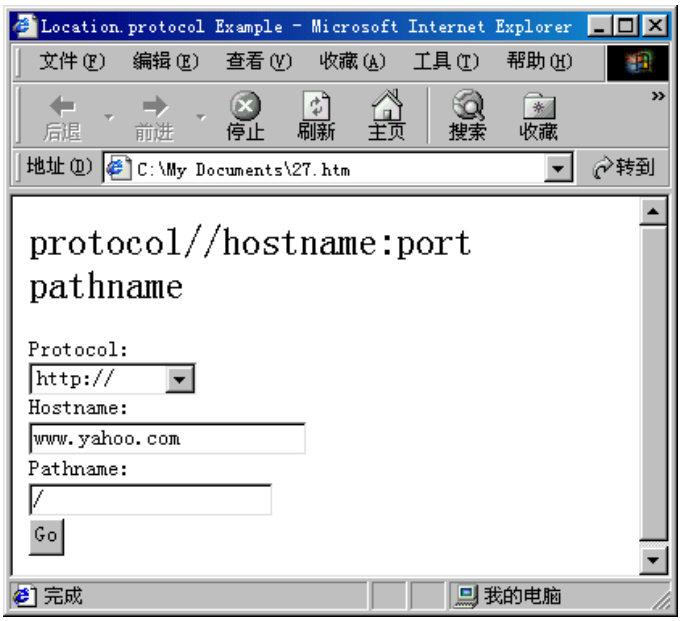


图9-43

第10章 history对象

主要内容

- history 对象的属性、方法

本章导读

本章主要通过两个具体的实例来介绍历史对象的 current、length、next、previous 属性以及 Back（）、Forward（）、Go（）方法的运用。

当我们在 Internet 上进行网上冲浪时，浏览器会自动维护一个用户最近访问的 URL 列表。这个列表是由 history 对象在 JavaScript 中完成的，一般来说在该列表中的 URL 是不能通过 Script 秘密获取的。

对于 history 对象及其它内部的 back（）或 go（）方法运用 HTML 文档中提供的 Back 按钮的替代功能。这个按钮触发了一个 Script，检查历史记录表中的所有元素，然后返回一个网页。您的文档不必知道任何关于用户是从哪里登录到您的 Web 页面的 URL 中的。

History 对象的主要属性如下：current、length、next、previous 等，主要方法如下：Back（）、Forward（）、Go（）等。

我们这里主要介绍 history 对象的 length 属性。用 history 对象的 length 属性来计算历史记录列表中的元素，但是这个有价值的信息在编写与当前位置相关的导航脚本中并不是十分有用，因为在脚本中不能从历史记录队列中当前文档所在处得到任何东西。但是如果当前文档在列表的顶端，那么我们可以计算文档的相对位置。

在下面的实例中我们将使用简单的函数来显示基于浏览器的历史记录中元素的个数和两个警告信息之一。下面是实例程序的代码清单：

```
<HTML>
<HEAD>
<TITLE>
 HISTORY 对象
</TITLE>
<SCRIPT LANGUAGE=”JavaScript”>
function showCount()
{
var hisCount=window.history.length
if (hisCount>5)
{
alert(“you have been busy, you have visited”+hisCount +”page so far”)
}
else
{
alert(“you have been to ”+gusCount +”Web pages this session.”)
}
}
</SCRIPT>
</HEAD>
<BODY>
<FORM>
<INPUT TYPE=”button” NAME=”activity” VALUE=”激活” onClick=”showCount()”>
</FORM>
</BODY>
</HTML>
```

在下面的实例中我们将利用 history 对象的常用属性、方法来创建一个 Web 页面。下面是实例的代码清单：

```
<html>
```

```
<head>
<title>History 对象</title>
</head>
<body bgcolor="#FFFFFF">
<SCRIPT LANGUAGE="JavaScript">
<!--
    function goPrev()
        parent.frames[1].history.back()
    }
    function goNext()
        parent.frames[1].history.forward()
    }
    function moveOn()
        var urlAddress = ""
        urlAddress = document.forms[0].LocationBox.value
        if (urlAddress != "")
                parent.frames[1].location = urlAddress
                document.forms[0].ListLen.value = parent.frames[1].history.length }
        else
                alert("请在单击按钮之前输入文本信息.")
        }
    }
    function jump()
        if (document.forms[0].goParam[1].checked)
                  var goVal = 0
                goVal = parseInt(document.forms[0].GoBox.value) }
        else
                var goVal = ""
                goVal = document.forms[0].GoBox.value
        }
        parent.frames[1].history.go(goVal)
    }
    function getParentInfo()
        myParentTitle = parent.document.title
        alert(myParentTitle)
    }
// -->
</SCRIPT>
<form method="POST">
<h3><em>Ways to navigate</em></h3>
<ul>
<li>使用 location 对象.</li>
</ul>
<p><input
    type=text
    size=20
    maxlength=50
```

```
    name="LocationBox"> <input
    type="button"
    value="查找"
    onClick="moveOn()"> </p>
<ul>
<li>使用 history 对象的 <em>back()</em> and<em> forward() </em>方法.</li>
</ul>
<p align=center><input
    type="button"
    value="&lt;&lt;"
    onClick="goPrev()"> <input
    type="button"
    value="&gt;&gt;"
    onClick="goNext()"></p>
<ul>
<li>使用 history 对象的 <em>go()</em> 方法:</li>
</ul>
<blockquote>
<p><input type=radio name="goParam" value="ByCount">By Count</p>
<p><input type=radio name="goParam" value="ByName">By Name</p>
<p><input
    type=text
    size=20
    maxlength=30
    name="GoBox"> <input
    type="button"
    value="Go"
    onClick="jump()"> </p>
<p>History 列表: <input
    type=text
    size=3
    maxlength=4
    name="ListLen"> </p>
</blockquote>
<input
    type="button"
    value="取得信息"
    onClick="getParentInfo()">
</form>
</body>
</html>
```

下面的代码是 Web 页面中的几个按钮的响应函数。它们分别使用了 history 对象的属性和方法。

```
<SCRIPT LANGUAGE="JavaScript">
<!--
    function goPrev()
        parent.frames[1].history.back()
    }
```

```
    function goNext()
        parent.frames[1].history.forward()
    }
    function moveOn()
        var urlAddress = ""
        urlAddress = document.forms[0].LocationBox.value
        if (urlAddress != "")
                parent.frames[1].location = urlAddress
                document.forms[0].ListLen.value = parent.frames[1].history.length }
        else
                alert("请在单击按钮之前输入文本信息.")
        }
    }
    function jump()
        if (document.forms[0].goParam[1].checked)
                  var goVal = 0
                goVal = parseInt(document.forms[0].GoBox.value) }
        else
                var goVal = ""
                goVal = document.forms[0].GoBox.value
        }
        parent.frames[1].history.go(goVal)
    }
    function getParentInfo()
        myParentTitle = parent.document.title
        alert(myParentTitle)
    }
// -->
</SCRIPT>
```

实例程序在浏览器中预览的效果如图 10-1 所示。

图10-44

第11章 layer对象

主要内容

- layer 对象的属性
- layer 对象的方法
- JavaScript 操作层

本章导读

在浏览器中，窗口中加载的主文档之上，每一层都有一些属于本层的几类 HTML 内容。临时更改或贴换单独一层的内容不会影响到其他层，且整个一层可以在 JavaScript 的控制下重定位、重设大小或隐藏属性。

11.30 layer 属 性

layer 对象主要具有如下的属性：above、background、below、bgcolor、clip、document、left、name、pagex、parentlayer、siblingBelow、src、top、visibility、zlndex 等。

在下面我们将分别介绍 layer 对象的一些常用的属性。

11.30.1 Above、below、siblingAbove、siblingBelow

每一层在它自己的物理层中都约定用传统的 x 和 y 变量代表宽和高，第三个尺度（层相对于层的堆叠的位置）叫做 z-order。层的顺序是根据加载的次序自动赋值的，最高的数目是顶层。

在页面中我们知道哪层在另一层的前面对我们编写 script 是很重要的，尤其是当您的 script 需要根据用户的动作重定位层的顺序时。Layer 对象提供了上面的四个属性来确定与某层相邻。

对于 layer.above 和 layer.below 两个属性，是全局地考虑页中定义的所有层，忽略一层内嵌套的层。如果一层位于某层的上面，那么就有属性指向该层的引用，如果在哪个方向没有层，则对该层引用的值就是空。例如在一个三层的文档中，第一层定义为最底层，它的上面有一层，则 layer.above 就有一个对上层的引用，但是没有下层，则 layer.below 的值就为空，我们在 Script 时就不能对 layer.below 属性引用，否则将导致运行时 script 出错，提示您的对象必须是有属性的。同理第三层就只具有下层，那么 layer.below 就有对下层的引用，而没有对上层的引用。

对于 layer.siblingAbove 和 layer.siblingBelow 两个属性是将它们自己限定在一个父层所包含的层组中。而实际就是在一个父层组内两个子层之间的引用。

11.30.2 background

在一个层中可以赋予一个背景图。初始图像一般都是由<LAYER>标记符的 BACKGROUND 属性设置的，但是在任何时候都可以通过 layer.background 属性赋予另外一幅新图像。

层的背景图就像整个网页中的图像，它们与页面中的内容不一样，同时它们也可以是一些内容，我们可以根据自己的需要选择适合自己的图像。

Layer.background 属性是一个图像对象。如果要在运行中改变图像，则必须将 layer.background.src 属性设置为期望的图像的 URL。如果调用的图像尺寸比层的区域小，则背景图像将自动复制来填充整个层区域的大小，而不是图像的缩放。当然我们也可以通过设置 layer.background.src 属性为空去掉层的背景图像。

下面我们将举一个简单的实例，在页面中定义了一层，该层主要有五个按钮来改变第二层的背景，同时将五个按钮放在一个层中是为了确定按钮和背景层的区域在所有的平台

都是按上边对齐。

当第二层加载时，只赋予一个灰色的背景色，写出一些反显的文本。这些图像都是由几幅风景图像所组成的。下面是实例程序的清单：

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE=”JavaScript”>
function setBg(URL)
{
document.bgExpro.background.src=URL
}
</SCRIPT>
</HEAD>
<BODY>
<B>Layer Background</B>
<HR>
<LAYER NAME=”buttons” top=50>
<FORM>
<INPUT TYPE=”button” CALUE=”First” onClick=”setBg(‘first.gif’)”>
<BR>
<INPUT TYPE=”button” CALUE=”Two” onClick=”setBg(‘two.gif’)”>
<BR>
<INPUT TYPE=”button” CALUE=”Three” onClick=”setBg(‘three.gif’)”>
<BR>
<INPUT TYPE=”button” CALUE=”Four” onClick=”setBg(‘four.gif’)”>
<BR>
<INPUT TYPE=”button” CALUE=”Five” onClick=”setBg(‘five.gif’)”>
<BR>
</FORM>
</LAYER>
<LAYER NAME=”bgExpro” BGCOLOR=”gray” top=50 left=250 width=300 height=250>
<b><FONT COLOR=”white”>which may or may not read well with the various backgrounds.click the button</FONT></B>
</LAYER>
</BODY>
</HTML>
```

11.30.3 Clip 属　性

Layer.clip 属性自身就是一个对象，它使用六个位置属性定义层中用户可见的矩形区域的位置和大小。这六个属性是：

- Clip.top
- Clip.left
- Clip.bottom
- Clip.right
- Clip.width

- Clip.height

这些属性的度量单位是像素，值是相对于层对象的左上角的。

剪裁区域的大小可以小于或等于层对象。如果在<LAYER>标记符中没有定义 Clip 属性，那么剪裁区域就与层同样大小。这样 clip.left 和 clip.top 值自动为零，因为剪裁区域从层的矩形区域的最左上角开始。

再设置一个 clip 属性不移动层或层的内容，而只是移动层的可见区域，每一次调整对可见区域的外观移动有独特地影响。例如，如果 clip.left 值从原来的 10 增加到 30，则整个矩形区域的左边就向右边移动 30 个像素，其他边保持不变。更改 clip.width 属性只移动右边；而改变 clip.height 属性只影响底边。不过，每次只能更改一边的属性。

在下面的实例中，我们将在页面中设计六个文本域来调整剪裁区域的大小。下面是实例程序的代码清单：

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE=”JavaScript”>
var origLayerWidth=0
var origLayerHeight=0
function initializeXY()
origLayerWidth=document.display.clip.width
origLayerHeight=document.display.clip.height
showValues()
}

function setClip(field)
{
var clipVal=parselnt(field.value)
document.display.clip[field.name]=clipVal
showValues()
}
function showValue()
{
var form=document.layer[0].document.forms[0]
var propName
for (var I=0;I<form.elements.length;I++)
{
propName=form.elements[I].name
if (form.elements[I].type= =”text”)
{
form.elements[i]value=document.display.clip[propName]
}
}
}
var intervallD
function revealClip()
{
var midWidth=Math.round(origLayerWidth/2)
var midHeight=Math.round(origLayerHeight/2)
```

```
document.display.clip.left=midWidth
document.display.clip.top=midHeight
document.display.clip.right=midWidth
document.display.clip.bottom=midHeight
intervallD=setlnterval(“stepClip()”,1)
}

function stepClip()
{
var widthDone=false
var heightDone=false
if (document.display.clip.left > 0)
{
document.display.clip.left+=-2
document.display.clip.right+=2
}
else
{
widthDone=true
}
if (document.display.clip.top > 0)
{
document.display.clip.top +=-1
document.display.clip.top +=1
}
else
{
heightDone=true
}
showValues()
if (widthDone && heightDone)
{
clearlnterval(intervallD)
}
}
</SCRIPT>
</HEAD>
<BODY onLoad=”initializeXY()”>
<B>layer clip 属性 </B>
<HR>
请输入 clip 各个边的属性值<p>
<LAYER TOP=80>
<FORM>
<TABLE>
<TR>
   <TD ALIGN=”right”>layer.clip.left：</TD>
   <TD><INPUT TYPE=”text” NAME=”left” SIZE=3 onChange=”setClip(this)”></TD>
```

```
</TR>
<TR>
   <TD ALIGN=”right”>layer.clip.top：</TD>
   <TD><INPUT TYPE=”text” NAME=”top” SIZE=3 onChange=”setClip(this)”></TD>
</TR>
<TR>
   <TD ALIGN=”right”>layer.clip.right：</TD>
   <TD><INPUT TYPE=”text” NAME=”right” SIZE=3 onChange=”setClip(this)”></TD>
</TR>
<TR>
   <TD ALIGN=”right”>layer.clip.bottom：</TD>
   <TD><INPUT TYPE=”text” NAME=”bottom” SIZE=3 onChange=”setClip(this)”></TD>
</TR>
<TR>
   <TD ALIGN=”right”>layer.clip.width：</TD>
   <TD><INPUT TYPE=”text” NAME=”width” SIZE=3 onChange=”setClip(this)”></TD>
</TR>
<TR>
   <TD ALIGN=”right”>layer.clip.height：</TD>
   <TD><INPUT TYPE=”text” NAME=”height” SIZE=3 onChange =”setClip(this)”></TD>
</TR>
</TABLE>
<INPUT TYPE =”button” VALUE=”更改设置” onClick=”revealClip()”>
</FORM>
</LAYER>
<LAYER NAME=”display” BGCOLOR=”coral” TOP=80 LEFT=200 WIDTH=360 HEIGHT=180>
<H2>看变换</h2>
<p>
请注意这里的变换效果
</p>
</LAYER>
</BODY>
</HTML>
```

实例是利用 JavaScript 编制的函数来缩放剪裁区域。剪裁区域的变换是通过四个边都设置到层的宽和高的中点来缩放。为了使得两个轴上的缩放都均匀，在两个方向都成比例地缩放，从而使剪裁区域的变换形成一幅动画的效果。

11.30.4 left、top 属　性

layer.left 和 layer.top 属性与<LAYER>标记符中的 LEFT 和 TOP 属性都相对应，这些值决定了层的左上角相对于浏览器窗口、框架或它所在的父层的水平和垂直的坐标。

调整这些属性都会重新定位层但不会改变其大小。剪裁区域的值不会因为这些属性的变换而改变。因此如果您要创建一个跟着在 X 和 Y 轴直线上拖动鼠标指针移动的可拖动的层对象，调整这些属性中的某个会比使用 layer.moveTo（）方法更方便。

11.30.5 Name 属　性

Layer.name 属性反应了<LAYER>标记符中的 NAME 属性或赋予给类似的 CSS-P 对象名字。该属性是只读的，如果在创建层时不赋予它一个名字，那么浏览器将自动产生一个名字形式的编号，但页面每次加载时将产生不同的编号，因此我们在设计程序时不能引用由浏览器自动产生编号，否则将发生错误。

11.30.6 src 属　性

层的内容可以从定义它的文档中来，也可以从外部资源中读取，如一个 HTML 或一个图像文件。如果<LAYER>属性定义由外部的文档来读取，则外部的文件由 SRC 属性指出。Layer.src 所反应的就是这个属性。

这个属性的值是一串外部文件的 URL。如果<LAYER>标记符中没有指定 SRC 属性，则返回空。不要将此属性设置为空来替代清除内容为 document.write（）的层或替代加载一个空页面。否则空字符串将被当成一个 URL 并加载到当前客户的浏览器中。

如果通过向层中加载一个新的源文件来改变层的内容，简单地将一个新的 URL 赋予 layer.src 属性即可。当然如果在层中嵌套有层，那么这些嵌套的层将被加载到正在改变 src 属性的层中的内容所替代。新文件是一个定义了自己的嵌套层的 HTML 文件。

11.31 layer 对象的方法

11.31.1 Load（“url”，newLayerWidth）方法

加载后改变层内容的一种方法是使用 layer.load（）方法。这个方法有利于设置 layer.src 属性，因为它的第二个参数就是新层内容的宽度。如果不指定第二个参数，将被一个小的缺省值替代。如果您认为新文档对于存在的层的高度太高，就需要在加载新文档前用 layer.resizeTo（）方法或设置独立的 layer.clip 属性。这使得层的可见区域保持合适的大小。

在下面的实例中我们将在页面中创建一个按钮来让用户在层中加入长短不同的文档。实例程序的代码清单如下：

```
<HTML>
<HEAD>
<SCRIPT>
function loadDoc(URL,width)
{
if (! width)
```

```
{
width=document.myLayer.clip.width
}
document.myLayer.load(URL,width)
}
</SCRIPT>
</DEAD>
<BODY>
<B>loading new documents</B>
<HR>
<LAYER TOP=90 WIDTH=240 BGCOLOR="yellow">
<FORM>
Loading new documents:
<BR>
<INPUT TYPE="button" VALUE="小文档" onClick="loadDoc('first.htm')">
<BR>
<INPUT TYPE="button" VALUE="大文档" onClick="loadDoc('two.htm')">
<P>
<INPUT TYPE="button" VALUE="小文档" onClick="loadDoc('first.htm',200)">
<BR>
<INPUT TYPE="button" VALUE="大文档" onClick="loadDoc('two.htm',200)">
<P>
<INPUT TYPE="button" VALUE="更新" onClick="loadDoc()">
</FORM>
</LAYER>
<LAYER NAME="mylayer" BGCOLOR="yellow" TOP=90 LEFT=300 WIDTH=300 HEIGHT=200>
<P>
<B>Text loaded in original document.</B>
</P>
</BODY>
</HTML>
```

11.31.2 MoveAbove(layerObject)、moveBelw(layerObject)方法

除了 layer.zIndex 属性外，层对象无法调整影响层的全局堆栈顺序的属性。Layer.moveAbove（）和 Layer.moveBelow（）方法可以调整与另外一层对象有关的层。这两个层必须是兄弟关系，也就是说他们必须包括在同层内。我们不能将已经存在的层从一个层中移动到另一个层中，而必须从源层中删除，然后在目的层中创建一个新层。这些方法都不会影响可见区域的大小和层的坐标系定位。

在方法中的参数是一个层对象的引用，即捕获的物理引用指针。例如：在下面的一段代码中：

document.fred.moveAbove(document.ginger)

指令是将 Fred 对象层移动到 ginger 对象层上面。Fred 层与 ginger 层的堆栈关系是任意的，但是它们必须嵌套在同一层中。

11.32 JavaScript 操作层

首先需要定义访问层的 JavaScript 的语法，因为对于不同的浏览器而有一些不同。例如 Navigator 和 Internet Explorer，他们将使用不同的方法来访问层。

在 Navigator 浏览器中是通过层数组来访问的，因为他们在层中定义了自己 name 属性，直接调用他们的 name 属性就可以访问层；而在 Internet Explorer 浏览器中是使用整个数组来访问的，在 name 属性中没有定义访问这个层的 name 属性，但是他们定义了一个叫 ID 的属性来访问层。

在下面的实例中我们将在页面中创建两个按钮和一个层。然后由这两个按钮来控制这个层的显示和隐藏。实例程序的代码清单如下：

```
<html>
<head>
  <style type="text/css">
  <!--
    #layer1{
    background-color: green;
    height: 100;
    left: 10;
    position: absolute;
    top: 50;
    width: 100;
    }
  -->
  </style>
  <script type="text/JavaScript" language="JavaScript1.2">
  <!--
    var isIE = new Boolean(false);
    var isNav = new Boolean(false);
    var unSupported = new Boolean(false);
    var layer = new String();
    var style = new String();
    function checkBrowser()
{
      if(navigator.userAgent.indexOf("MSIE") != -1){
        isIE = true;
        layer = ".all";
        style = ".style";
      }
else if(navigator.userAgent.indexOf("Nav") != -1){
        isNav = true;
        layer = ".layers";
        style = "";
```

```
        }
else
{
            unSupported = true;
        }
    }
    function changeState(layerRef, state){
        eval("document" + layer + "['" + layerRef + "']" + style + ".visibility = '" + state + "'");
    }
  //-->
  </script>
</head>
<body onload="checkBrowser()">
  <div name="layer1" id="layer1">
    DIV 1
  </div>
  <form name="form1">
    <input type="button" value="Hide" onclick="changeState('layer1','hidden')">
    <input type="button" value="Show" onclick="changeState('layer1','visible')">
  <form>
</body>
</html>
```

实例在浏览器中预览的效果如图 11-1 所示。

图11-45

第12章 字符串对象

主要内容

- 转 义 字 符
- 字符串对象的属性
- 字符串对象的方法

本章导读

该对象是为一系列用双引号或单引号定义的字符。例如: string= “这是一个字符串对象”。在设置字符串对象的时候一定要注意引号的匹配。

12.33 转义字符

在 JavaScript 中，有一些特殊的字符串对象，主要是引号、回车符号、空格键等等。下面列出这些字符的实现方法。

双引号，\”

单引号，\’

反斜杠，\\

退格，\b

Tab，\t

换行，\n

回车，\r

进格，\f

例如下面的语句就可实现以上的特殊字符，在编写 JavaScript 程序中，通过反斜杠加上特殊的字符就实现了特殊字符在页面中表现的方法。请看下面的示例，示例效果如图 12-1 所示。

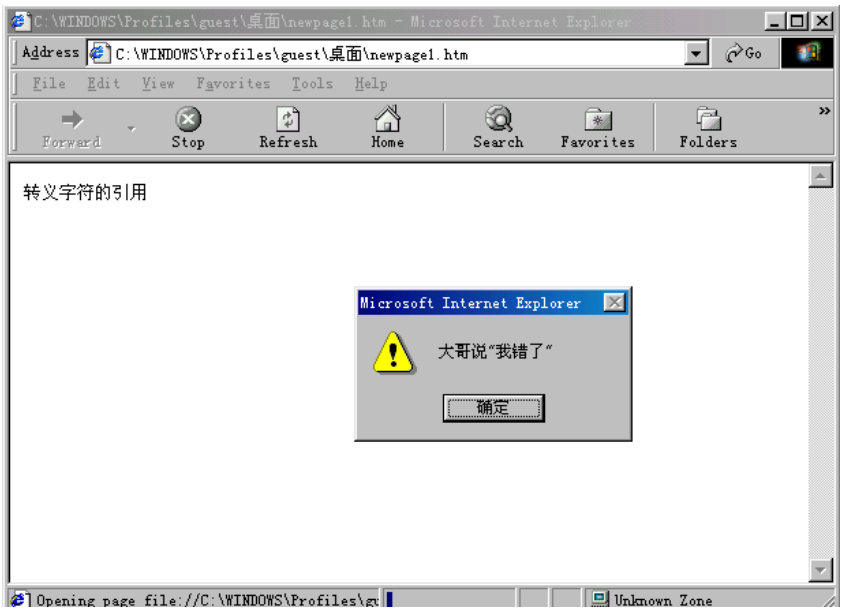


图12-46

原代码如下：

```
<html>

<head>
<title></title>
</head>

<body>
```

```
<p align="left">转义字符的引用  </p>
<script language="JavaScript">
<!--
yuju="大哥说\"我错了\""
alert(yuju)
// -->
</script>

</body>
</html>
```

其中：

- **yuju="大哥说\"我错了\""**：该语句设置一个字符串对象;
- **alert(yuju)**：调用 window 的 alert 方法，在文档窗口显示一个提示框，在提示框中显示设置 “大哥说"我错了"” 这样的字符。

通过该例的设置，我们对字符串对象有了一个大致的了解，下面我们来看看该对象的属性、方法以及具体的设置方法。

12.34 字符串对象的属性

length 属性用来测量字符串对象的长度。在实际的程序设计中，一般用它来获得用户输入数据的长度。在以后的数据校验以及数据的判断中，都要利用到该属性。

该属性的引用格式一般为：

string.length

看下面的例子，我们来设置该属性。

```
<html>

<head>
<title>C：\My Documents\zhuanyi.htm</title>
</head>

<body>

<p align="left">转义字符的引用  </p>
<script language="JavaScript">
<!--
yuju="大哥说\"我错了\""
cd=yuju.length
document.write(cd)
wo="hello，world"
js=wo.length
document.write("<br>"+js)
```

```
// -->
</script>

</body>
</html>
```

在 IE 中浏览的效果如图 12-2 所示。

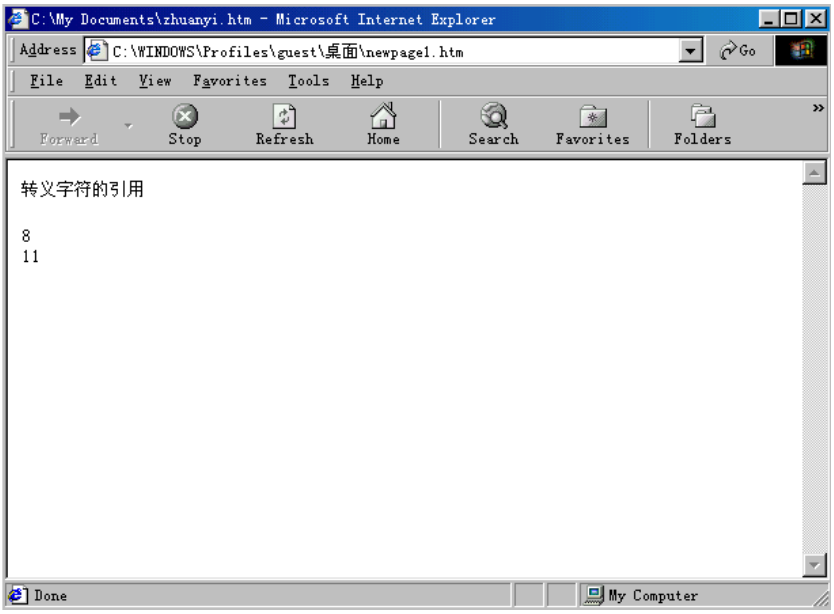


图12-47

其中：

- **wo="hello，world"**：定义一个字符串对象；
- **js=wo.length**：设置变量，获得字符串对象的长度值；
- **document.write("
"+js)**：在文档中写出变量的值，该值就是设置的字符串对象的 length 属性值。

12.35 字符串对象的方法

12.35.1 “+”加 法

该方法将两个字符串连接起来，在实际的应用中常涉及到。请看下边的例子。

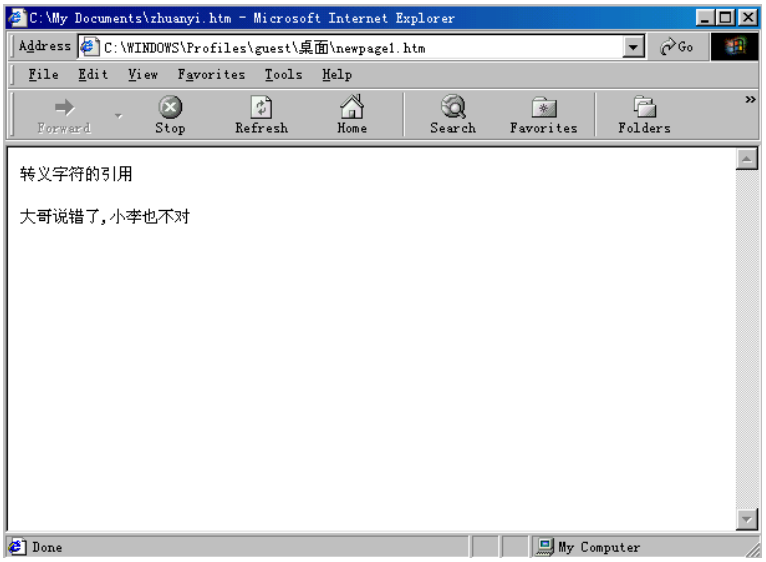
```
<html>

<head>
<title>C：\My Documents\zhuanyi.htm</title>
</head>
```

```
<body>

<p  align="left">转义字符的引用
</p>
<script language="JavaScript">
<!--
yuju="大哥说错了，"
cd="小李也不对"
document.write(yuju+cd)
// -->
</script>

</body>
</html>
```



在浏览器中，该页面如图 12-3 所示，可以看出，前后两个字符串对象被连起来了。

图 12-3

12.35.2 charAt

该方法从字符串中返回一个字符，这个方法在应用的时候，通常会设置一个起始位置的参数，然后传回位于该字符串对象位置的字符值。在使用的时候，系统认为字符串起始的位置为 0。

该方法的调用方式如下：

string.charat（index）

其中：

- **index**：就是用户设置的参数，用来获得字符的位置。

请看下面的例子：

```
<html>

<head>
<title>My Document</title>
</head>

<body>

<p align="left">转义字符的引用  </p>
<script language="JavaScript">
<!--
yuju="大哥说错了"
cd=yuju.charAt(2)
document.write(cd)
// -->
</script>
```

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

其中：

- **yuju="大哥说错了"**：设置字符串对象；
- **cd=yuju.charAt(2)**：在该语句中，调用字符串对象的 charAt 方法，获得该字符串对象的第三个位置的值，即“说”，然后将该字符赋给设置的变量 cd；
- **document.write(cd)**：在文档中写出设置的变量值。

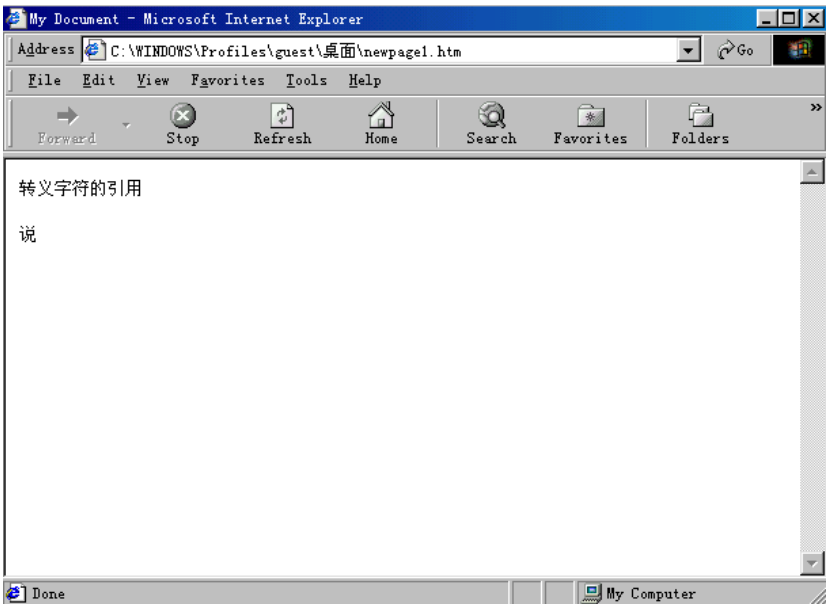


图 12-4

在浏览器中，该页面显示的效果如图 12-4 所示。

12.35.3 indexOf

该方法从一个特定的位置开始查找设置的字符、返回一个字符或是字符串上起始位置值，如果在特定的位置找不到用户设置的字符串对象，则返回一个-1，利用这样的特性，在利用 JavaScript 设置查找数据的格式化过程中非常有用。在后面的综合部分中，系统常利用该方法来设置检索和数据检验方面的工作。这里只简单地介绍一下其用法。

该方法的调用格式如下：

string.indexOf(string，index)

其中：

- **括号中 string**：指的是要查找的字符串；
- **index**：指的是起始位置。

看下面的示例：

```
<html>

<head>
<title>C：\My Documents\zhuanyi.htm</title>
</head>

<body>

<p align="left">转义字符的引用  </p>
<script language="JavaScript">
<!--
yuju="字符串对象"
```

```
cd="对象"
re=yuju.indexOf(cd)
document.writeln(re)
ls=yuju.indexOf(cd，5)
document.write("<br>"+ls)
// -->
</script>

</body>
</html>
```

其中：

- **yuju="字符串对象"**：设置字符串对象；
- **cd="对象"**：设置查找的字符；
- **re=yuju.indexOf(cd)**：该语句用来在设置的字符串对象 yuju 中查找“对象”二字，如果查找到了，就将该字符位于字符串对象的起始位置赋给设置的变量“re”，这里通过分析，可以得出 re 的值为“3”；
- **document.writeln(re)**：在文档中写出变量的值；
- **ls=yuju.indexOf(cd，5)**：设置查询的起始位置为第五个字符处，该字符的长度为 5，所以该对象返回的值为-1，即 ls 的值为-1。

在浏览器中，该页面的显示如图 12-5 所示。

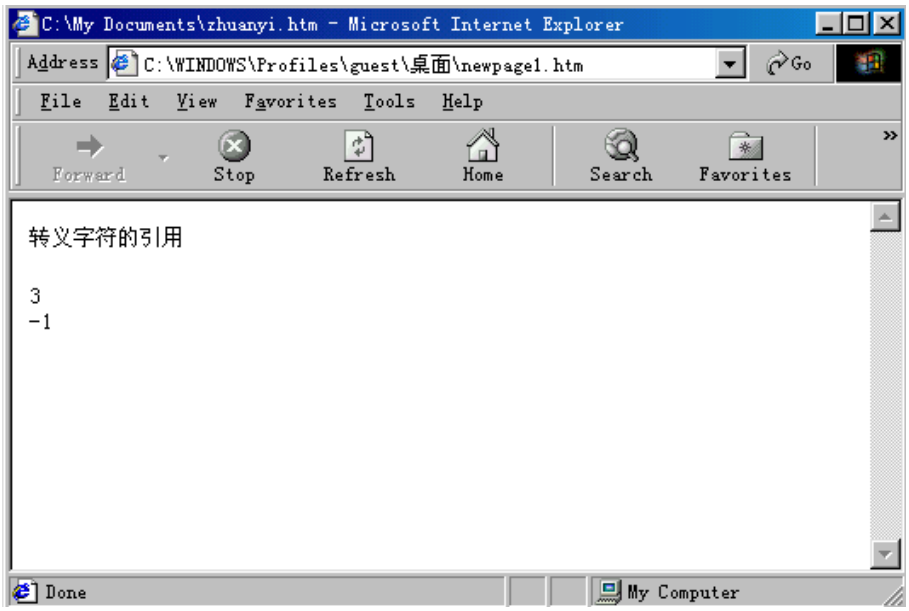


图 12-5

12.35.4 lastIndexOf

该方法的使用与上面的 indexOf 的使用方法一样，只是该方法从字符串对象的尾部开始搜索。我们将上面的例子稍加改动，看看这两个方法的不同处。

```
<html>

<head>
<title>C：\My Documents\zhuanyi.htm</title>
</head>

<body>

<p align="left">转义字符的引用　</p>
<script language="JavaScript">
<!--
yuju="字符串对象"
cd="对象"
re=yuju.lastindexOf(cd)
document.writeln(re)
// -->
</script>

</body>
</html>
```

在浏览器中，该页面显示的值为 3，如图 12-6 所示。

图 12-6

12.35.5 substring

该方法为字符串截取方法，在设置的时候，一般会设置两个参数来指定截取的位置，然后将两个参数间的字符串返回给设置的变量。当两个参数相等的时候就返回一个空字符串。在设计的运行中，一般可以不管参数的大小和前后的位置，它截取的顺序都是从最小的参数开始，截取到最大参数位置。如果没有指定最后的参数，系统默认截取到字符串对象的末尾。

该方法的调用格式如下：

string.substring(index1，index2)

其中：

- Index1 和 index2 中较小的一个为截取的起始位置。

请看下边的设置示例：

```
<html>

<head>
<title>C：\My Documents\zhuanyi.htm</title>
</head>
<body>

<p align="left">转义字符的引用　</p>
<script language="JavaScript">
<!--
yuju="字符串对象"
re=yuju.substring(3，5)
document.writeln(re)
// -->
</script>

</body>
</html>
```

其中：

- **yuju="字符串对象"**：设置字符串对象;
- **re=yuju.substring(3，5)**：调用 substring()方法，从字符串对象的第三个位置截取到第五个位置，然后将截取的对象赋给设置的变量 re，这里 re 的值为“对象”；
- **document.writeln(re)**：在文档中写出变量的值。

在浏览器中，该脚本的情况如图 12-7 所示。

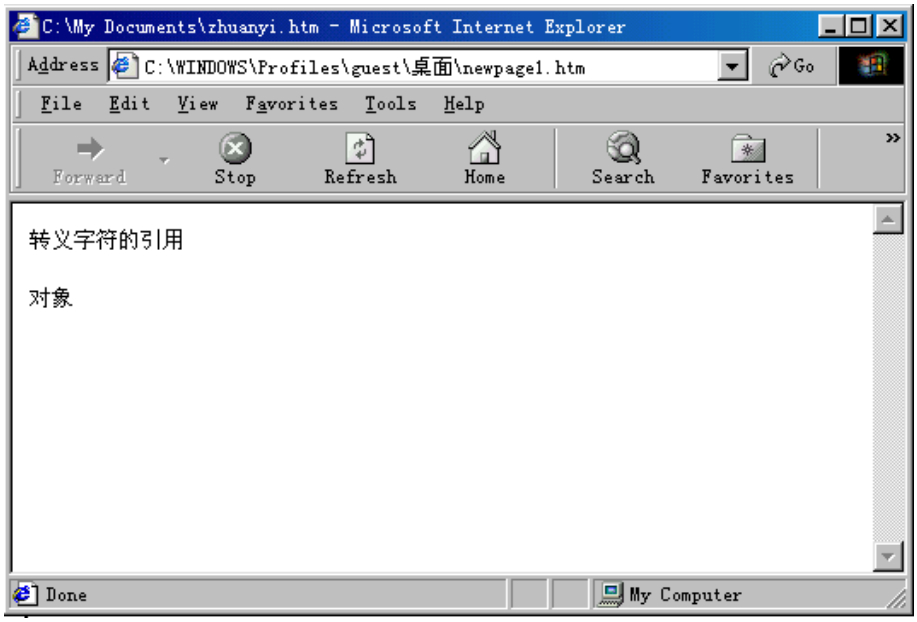


图 12-7

12.35.6 toLowerCase

该方法将字符串对象中的所有字符转换为小写的字符。在利用该方法后，成批次地将用户设置的字符串对象转换为小写字符。在转换的过程中，文本字符串对象不会发生质的转变。

该方法的调用格式如下：

string.toLowerCase()

其中：

- string 指的是用户设置的字符串对象。

请看下面的例子：

```
<html>

<head>
<title>xiaoxie</title>
</head>

<body>

<p align="left">转义字符的引用  </p>
<script language="JavaScript">
<!--
yuju="THIS IS A BIG DOG"
document.writeln("大写：")
document.writeln(yuju)
re=yuju.toLowerCase()
document.writeln("<br>小写："+re)
// -->
</script>

</body>
</html>
```

其中：

- **yuju="THIS IS A BIG DOG"**：设置字符串对象。由于在设置的过程中，举例设置转换大写字符为小写字符，所以这里设置的是大写的字符串对象;
- **document.writeln("大写：")**：该语句将设置的字符串对象在文档中写出来，并冠以大写的提示;
- **re=yuju.toLowerCase()**：调用设置的方法，将设置的字符串转换为小写字符;
- **document.writeln("
小写："+re)**：在文档中将转换而来的小写字符串对象写出来。

该脚本在浏览器中的显示情况如图 12-8 所示。

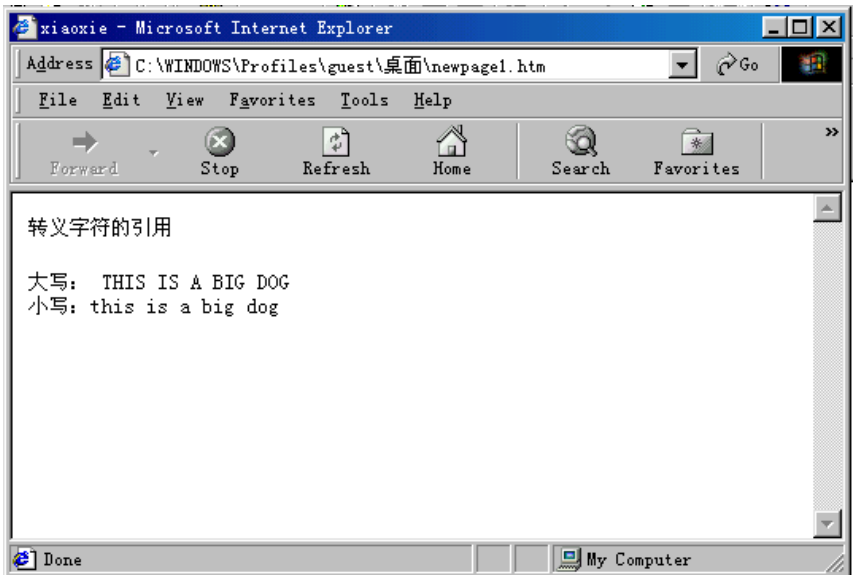


图 12-8

12.35.7 toUpperCase

该方法同上，只是将字符串对象中的小写字符转换为大写字符。该方法的调用格式同上。

这两种方法将字符串对象转换为用户指定的格式，在实际的脚本设计中主要是利用该方法来校验用户的输入，比如确定用户输入的密码，指定用户名称等等。

请看下面的例子。

```
<html>

<head>
<title>daxie</title>
</head>

<body>

<p align="left">转义字符的引用  </p>
<script language="JavaScript">
<!--
yuju="THIS IS A BIG DOG"
document.writeln("大写：")
document.writeln(yuju)
re=yuju.toLowerCase()
document.writeln("<br>小写："+re)
document.writeln("<br>大写：")
result=re.toUpperCase()
document.writeln(result)
```

```
// -->
</script>

</body>
</html>
```

该例子在上面例子的基础上，设置了将转换而来的小写字符串对象转换为大写字符串。

其中：

- **result=re.toUpperCase()**：这里 re，就是利用 toLowerCase 转换而来的小写字符串对象。该语句利用设置的方法将字符串对象全部转换为大写的字符串对象;
- **document.writeln(result)**：该语句在文档中显示用户设置的字符串对象。

在浏览器中，该页面的显示情况如图 12-9 所示。

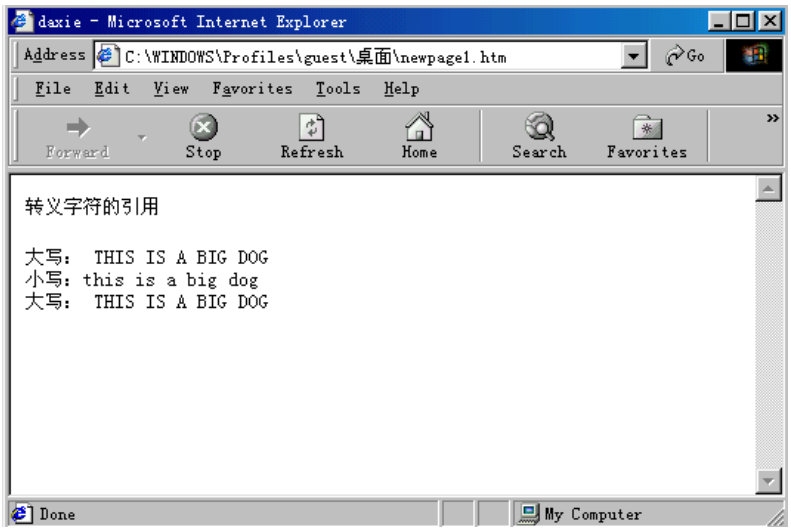


图 12-9

12.35.8 anchor

利用该方法在页面中创建和显示一个 HTML 超文本目标。在利用该方法的时候，必须在文档中建立一个锚点，然后调用 write 方法在文档中写出该连接锚点，达到在页面中快速定位的目的。

该方法的调用格式如下：

string.anchor(anchorname)

其中：

- **string**：指的是要设置锚点的字符串对象;
- **anchorname**：系统设置的锚点。

下面的实例讲解该方法的利用。

```
<html>

<head>
<title>daxie</title>
```

```
</head>

<body>

<p align="left">字符的<a href="#anc">引用</a>  </p>
<script language="JavaScript">
<!--
zifu="this ia anchor"
anc=zifu.anchor("anc")
document.write(anc)

// -->
</script>

</body>
</html>
```

其中：

- **zifu="this ia anchor"**：设置字符串对象；
- **anc=zifu.anchor("anc")**：设置 anchor 名称；
- **document.write(anc)**：在文档中设置该锚点。
- **字符的引用**：设置在页面中应用该锚点。在用户的利用中，只要点击该连接，就会跳到设置的锚点位置。

该脚本在浏览器中的显示情况如图 12-10 所示。

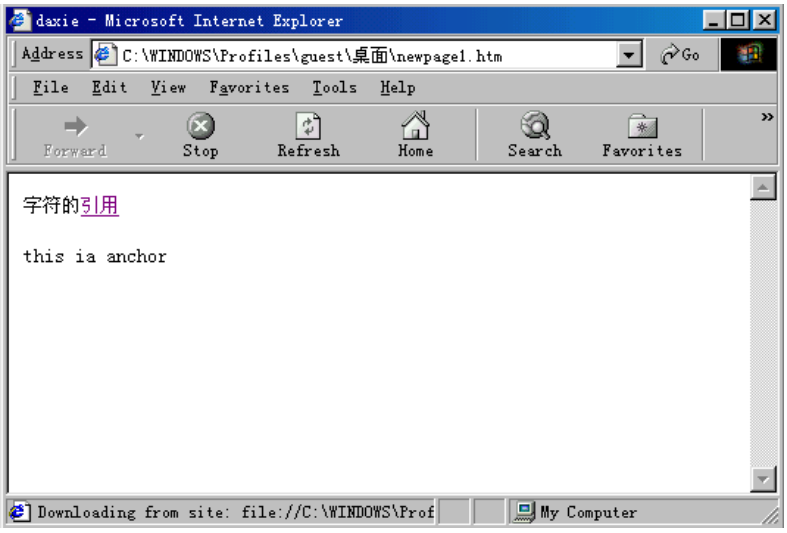


图 12-10

12.35.9 big

该方法将设置的字体转换为大字体格式，在功能的实现上，跟 HTML 语言的<big>一

样。在页面中显示时，会将设置的字符串对象大号显示。该方法的应用格式如下：

```
<html>

<head>
<title>daxie</title>
</head>

<body>

<p align="left">字符的引用　</p>
<script language="JavaScript">
<!--
zifu=" 您好"
anc=zifu.big()
document.write(anc)

// -->
</script>

</body>
</html>
```

其中：

- **zifu=" 您好"**：设置字符串对象；
- **anc=zifu.big()**：调用系统设置的方法将字体放大显示；
- **document.write(anc)**：在文档中显示用户的设置。

在 IE 中浏览该页面的情况如图 12-11 所示。

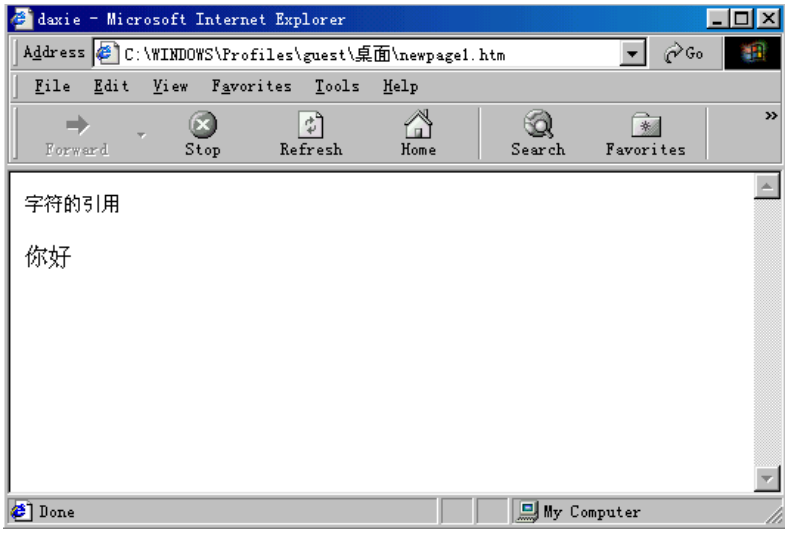


图 12-11

注意 与上面方法功能设置相似的还有：string.small()。

12.35.10 bold

该方法将设置的文本粗体显示。在功能的实现中与 HTML 标记中的<b>相似。这些方法实质上是 JavaScript 格式的控制语言。

该方法的应用格式如下：

string.bold()

其中：

- **string**：设置字符串对象。

该方法在实际的页面设计中的应用情况如下：

```
<html>

<head>
<title>daxie</title>
</head>

<body>

<p align="left">字符的引用  </p>
<script language="JavaScript">
<!--
zifu="字符的引用"
anc=zifu.bold()
document.write(anc)

// -->
</script>

</body>
</html>
```

其中：

- **zifu="字符的引用"**：用户定义的字符串对象;
- **anc=zifu.bold()**：调用系统设置的方法，将字符串对象转换为粗体的格式;
- **document.write(anc)**：在文档中将设置的格式显示出来。

在浏览器中，该脚本的表现情况如图 12-12 所示。

该方法与功能相似的地方如下：

- **string.blink()**：设置字体的闪烁;
- **string.itaics()**：设置文本的斜体显示;
- **string.strike()**：设置文本下划线;
- **string.sub**：设置下标;
- **string.sup**：设置上标;
- **string.fontsize**：设置字体的大小;
- **string.fontcolor**：设置字体的颜色。

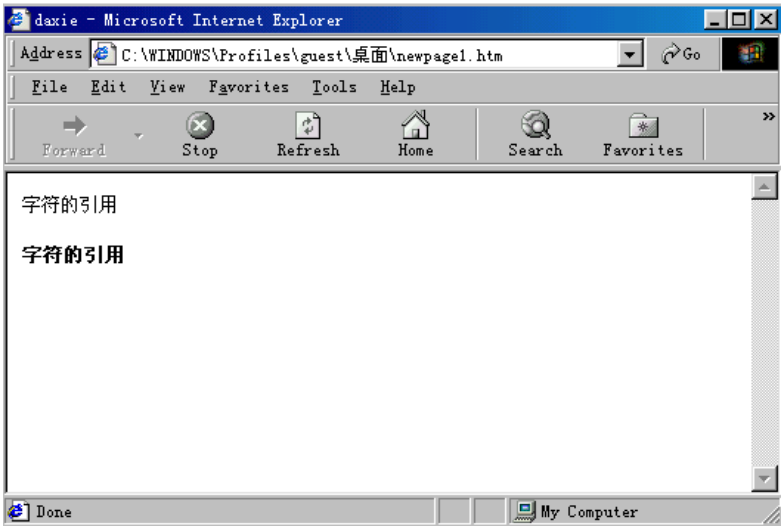


图 12-12

在如下的例子中，系统地看看这些格式的设置情况：

```
<html>

<head>
<title>daxie</title>
</head>

<body>

<p align="left">字符的引用　</p>
<script language="JavaScript">
<!--
zifu="字符的引用"
anc1=zifu.bold()
document.write(anc1)
anc2=zifu.italics()
document.write("<br>"+anc2)
anc3=zifu.sub()
document.write("<br>"+anc3)
anc4=zifu.sup()
document.write("<br>"+anc4)
anc5=zifu.fontsize(7)
document.write("<br>"+anc5)
anc6=zifu.fontcolor("red")
document.write("<br>"+anc6)

// -->
</script>

</body>
</html>
```

在该脚本的设置中，设置了一个字符串对象，然后调用设置的字符串对象方法，用户设置的字符串对象转换为相应的格式。

其中：

- **anc5=zifu.fontsize(7)**：设置字体的大小。后设置的参数数字一般为 1-7;
- **anc6=zifu.fontcolor("red")**：设置字体的颜色，这里可以直接是用户设置的色彩值。

在浏览器中，页面的显示情况如图 12-13 所示。

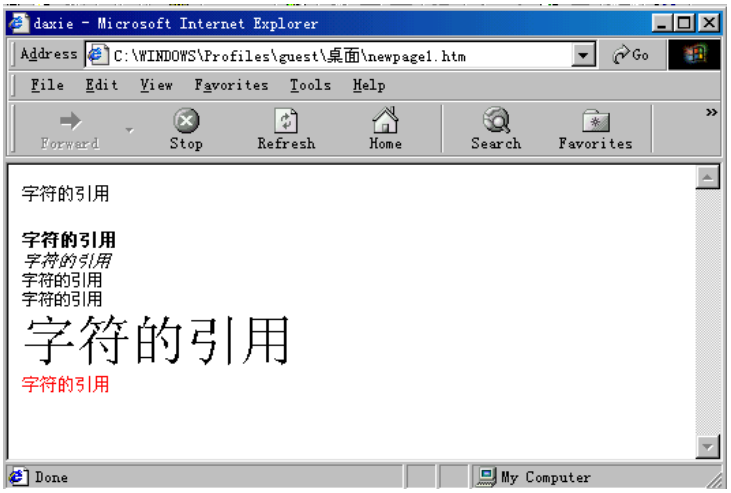


图 12-13

第13章 日期对象

主要内容

- 时间对象的属性
- 时间对象的方法

本章导读

本章主要通过具体的实例来介绍时间对象的设置以及该对象的 setYear、setTime、getTime、getDay、getDate、getMonth、getYear 等方法的运用。

13.36 时间对象的属性

在 JavaScript 中，提供了一些处理日期的对象和方法。通过 JavaScript 日期对象的帮助，您可以提取系统内部的时间，设置新的时间。通常，我们会在一些页面上看到用户设置的时钟，它给人一种非常专业的感觉。当您在页面中设置了一些分时的问候，中文的星期格式或者是动态的时钟的格式的时候，您一定会对页面设计有更深入的理解，一定对 JavaScript 脚本编程有了理性的认识，这样也深入 JavaScript 程序设计了。

在 JavaScript 中，将 date 作为一个对象来理解时，用户还需在使用的时候加以定义，先定义一个日期对象实体，一般就将 Date 认为一个时间关键字。在用户定义了用户的时间对象之后，就可以设置系统的时间，调用系统预定义的方法，设置用户想要设置的时间效果，开发有关时间的系统了。

日期对象属于 JavaScript 的内建对象。该对象没有任何属性，但系统设置了一些方法来实现时间的设置、修改。

在系统中，虽然我们看到的是实际时间的显示，但是系统存储的时间却是 1970 年 1 月 1 日午夜算起的毫秒数。所以在日期对象中禁止使用 1970 年 1 月 1 日前的时间。下面我们来看看时间对象的设置、时间对象方法的操作。

13.37 时间对象的设置

13.37.1 NEW

在 JavaScript 中，日期对象的设置要利用到一个关键字 new。这个 new 关键字的运用在 JavaScript 中相当有用，对于新对象的创建都要利用到该关键字。对于日期对象的创建的格式如下：

thisday=new Date（）.

其中：

- thisday 是我们新创建的日期对象。

对于新创建的日期对象，我们就可以在脚本设计中直接应用了。如果要日期对象跟上相应的参数，那么其格式如下：

thisday=new Date(month day,year　hours:minutes:seconds)

其中：

- **month**：设置新日期的月份;
- **day**：设置新日期对象的日子;
- **year**：设置新日期对象的年份;
- **hours**：设置新日期对象的小时数;

- **minutes**：设置新日期对象的分钟数;
- **seconds**：设置新日期对象的秒数。

对于新设置的时间数，系统认为该设置是可以选择设置的，也就是说，该日期的时间可以设置也可以不设置。

请看下边的实例。

```
<html>
<head>
<title></title>
</head>

<body>

<p><script language="JavaScript">
<!--
thisday=new Date()
document.write(thisday)
thisday2=new Date("08/09/1999")
document.write("<br>"+thisday2)
thisday3=new Date("08/09/1999 20:15:15")
document.write("<br>"+thisday3)
// -->
</script> </p>
</body>
</html>
```

在本例中，设计了设置日期对象的实例。在脚本的设计中，一共设置了三个时间对象来处理日期对象设置的三个方面。一个是取得用户当前系统的时间，另两个是取得用户设置的时间，一个设置没有跟时间，只有日期，一个设置了时间，最后调用文档对象的 write（）方法将用户设置的日期对象显示出来。通过该实例的设置，用户可以熟悉日期对象设置的各种情况。

在该实例中：

thisday=new Date()，设置一个日期对象，在该设置中，没有跟任何的参数。这样的设置，系统默认为取得用户当前日期。

document.write(thisday)，该语句调用文档对象的 write()方法，将用户设置的日期在文档中显示出来。

thisday2=new Date("08/09/1999")，设置另一个日期对象，设置的日期为 1999 年 8 月 9 日。这样的设置系统默认的时间为 0 点 0 分 0 秒。

document.write("
"+thisday2)，在文档中显示设置的日期变量 2。

thisday3=new Date("08/09/1999 20:15:15")，设置一个日期对象，该对象的设置中加入了系统设置的时间 20 点 15 分 15 秒。用户可以比较它和上一个日期对象的不同，在系统显示的时候，就会显示用户设置的时间而不是系统指定的时间 0 点 0 分 0 秒。

在浏览器中，脚本的执行情况如图 13-1 所示。

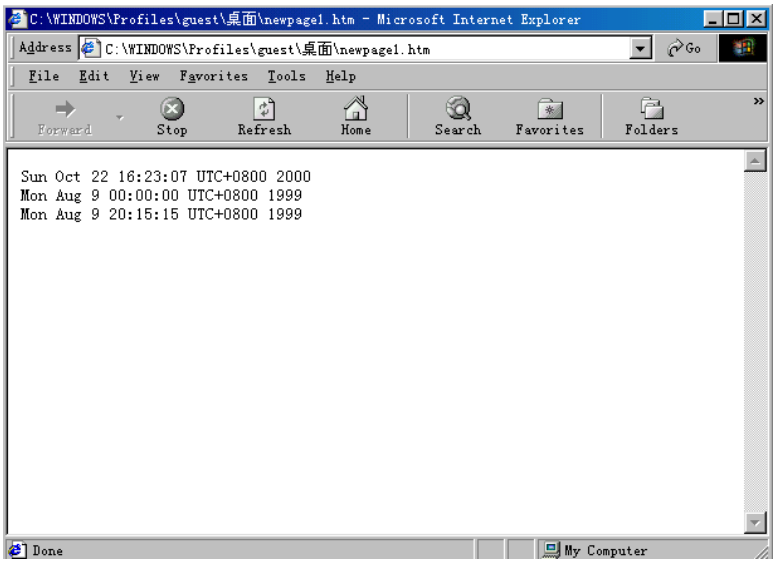


图13-48

下面来看看日期对象方法的应用。

13.37.2 getYear

该方法获得用户设置的日期对象值与 1900 的差值。尽管用户在设置日期对象的时候是向日期对象传递了一个四位的年份值，但是利用该方法获得的年份值却为一个两位数。在页面设计中，可以将设置的值用语句返回一个四位的数字。

该方法的应用格式如下：

dateobj.getYear()

其中：

- **dateobj**：是用户设置的日期对象。

在脚本设计中，该格式的引用如下：

```
<html>

<head>
<title></title>
</head>

<body>

<p><script language="JavaScript">
<!--
thisday=new Date("08/09/1999")
thisyear=thisday.getYear()
thisyear1=thisyear+1900
document.write("<br>"+thisyear)
document.write("<br>设置的日期年份为："+thisyear1+"年")
// -->
</script> </p>
</body>
</html>
```

在该脚本中，设置日期对象的年份为 1999 年 8 月 9 日而没有设置时间值，然后利用 getYear 方法获得用户设置的年份值，然后将获得年份值赋给设置的变量 thisyear，然而该值为一个与 1900 相比较带来的两位数，然后在设置与 1900 相加，将年份值转换为四位的

格式。

其中：

- **thisday=new Date("08/09/1999")**：设置日期对象;
- **thisyear=thisday.getYear()**：取得用户设置的日期对象的年份值;
- **thisyear1=thisyear+1900**：设置年份为四位数的格式;
- **document.write("
"+thisyear)**：显示两位格式;
- **document.write("
设置的日期年份为： "+thisyear1+"年")**：显示四位格式的设置。

在浏览器中，脚本运行的情况如图 13-2 所示。

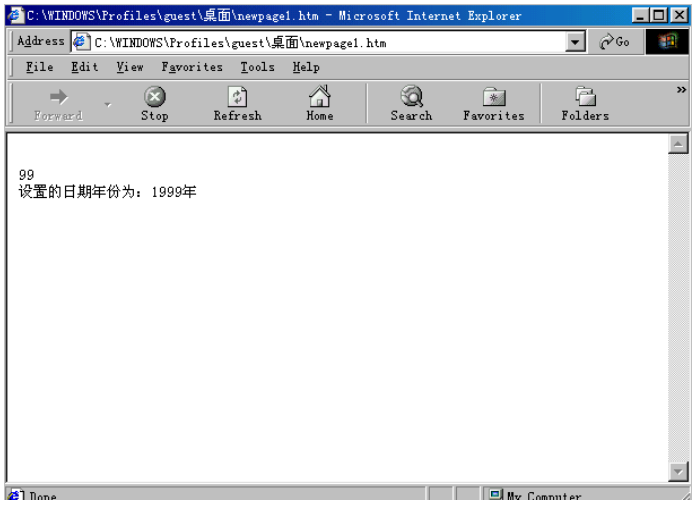


图13-49

13.37.3 getMonth

该方法获得设置日期对象中的月份值。如果月份为 1 月，则返回一个零值，如果为 12 月则返回一个 11，所以该方法返回的值为 0-11，在具体的设计中，还可在返回值的基础上加上 1，才能获得系统设置的月份。该方法的调用格式同上。请看下边的实例。

```
<html>
<head>
<title>
</title>
</head>
<body>

<p>
<script language="JavaScript">
<!--
thisday=new Date()
thismonth=thisday.getMonth()
thismoth=thismonth+1
```

```
document.write("<br>系统当前的月份为："+thismoth+"月")
// -->
</script>
</p>
</body>
</html>
```

该脚本的设计获得系统当前日期的月份。在该设置的过程中，设计了一个转换过程，由于系统当前的月份与利用该方法获得值相差一个 1，所以在具体的设置过程中，要在获得值中加上 1。

其中：

- **thisday=new Date()**：设置日期对象，由于没有跟任何的参数，所以系统默认是取得系统当前的日期;
- **thismonth=thisday.getMonth()**：调用设置的方法，获得系统的月份值。该值与系统当前的月份相差 1;
- **thismoth=thismonth+1**：设置转换，将获得的值加上 1;
- **document.write("
系统当前的月份为："+thismoth+"月")**：在文档中写入系统当前的月份。

在浏览器中，该脚本的执行情况如图 13-3 所示。

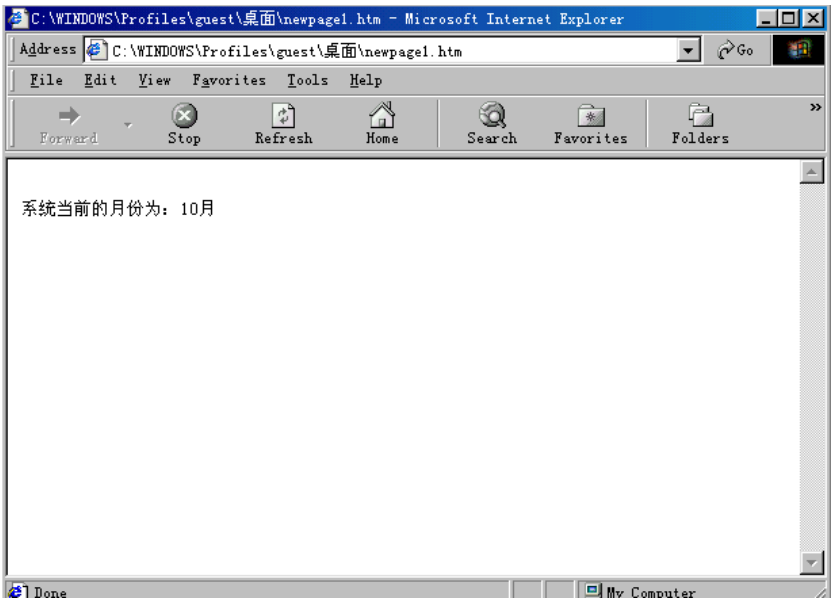


图13-50

13.37.4 getDate

该方法返回一个月份中的日期值。这个方法直接返回一个 1 到 31 之间的日期值，它和其它的设计有了较大的区别，所以在实际的设置中，可以直接应用该方法获得值。

该方法的调用格式如下：

dateobj.getDate()

在页面设置中，应用的格式如下：

```
<html>

<head>
<title></title>
</head>

<body>

<p><script language="JavaScript">
<!--
thisday=new Date()
thismonth=thisday.getMonth()
thismoth=thismonth+1
document.write("<br>系统当前的月份为："+thismoth+"月")
thisdate=thisday.getDate()
document.write("<br>系统当前的日期为："+thisdate+"号")
// -->
</script> </p>
</body>
</html>
```

其中：

- **thisday=new Date()**：设置系统当前的日期;
- **thismonth=thisday.getMonth()**：取得用户设置的日期对象的月份值;
- **thismoth=thismonth+1**：月份值的转换;
- **document.write("
系统当前的月份为："+thismoth+"月")**：显示系统设置的月份;
- **thisdate=thisday.getDate()**：取得当前日期位于该月的天数;
- **document.write("
系统当前的日期为："+thisdate+"号")**，在文档中显示该日期值。

在浏览器中，该页面的执行情况如图 13-4 所示。

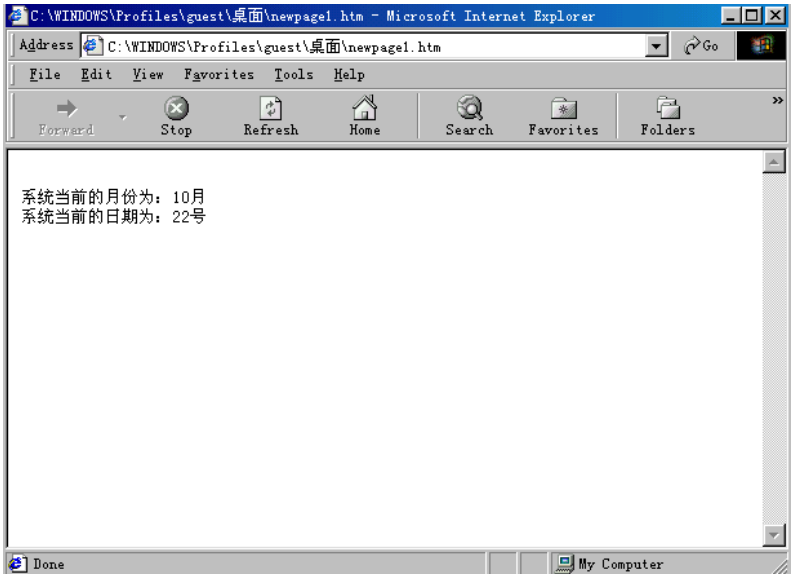


图13-51

13.37.5 getDay

该方法取得用户设置的日期的星期数。系统利用该方法获得值为 0 到 6。其中 0 表示星期天，而 6 表示星期六。所以在设置的过程中，除了 0 不符和国人的习惯外，其它的都可以直接引用。

该方法的调用格式如下：

dateobj.getDay()

在如下的实例中，设置了中文星期的显示，其源代码如下：

```
<html>

<head>
<title></title>
</head>

<body>

<p><script language="JavaScript">
<!--
thisday=new Date()
week=thisday.getDay()
document.write("<br>系统当前的星期数为："+week)
if(week==0)
document.write("<br>今天星期天")
if(week==1)
document.write("<br>今天星期一")
if(week==2)
document.write("<br>今天星期二")
if(week==3)
```

```
document.write("<br>今天星期三")
if(week==4)
document.write("<br>今天星期四")
if(week==5)
document.write("<br>今天星期五")
if(week==6)
document.write("<br>今天星期六")

// -->
</script> </p>
</body>
</html>
```

该脚本为一个范例，用户在设计页面脚本的过程中，可以直接调用。该脚本的设计就那么几句话，即可实现日期的中文显示。其中：

- **thisday=new Date()**：设置日期对象，该对象为系统当前的时间;
- **week=thisday.getDay()**：设置当前日期的星期数;
- **document.write("
系统当前的星期数为： "+week)**：显示系统当前的星期数。由于该日期为一个数字。所以在如下的语句中设置了一系列 if 语句，获得系统星期数的中文格式，并在文档中显示系统当前的星期数。

```
if(week==0)
document.write("<br>今天星期天")
if(week==1)
document.write("<br>今天星期一")
if(week==2)
document.write("<br>今天星期二")
if(week==3)
document.write("<br>今天星期三")
if(week==4)
document.write("<br>今天星期四")
if(week==5)
document.write("<br>今天星期五")
if(week==6)
document.write("<br>今天星期六")
```

在浏览器中，该脚本执行后，就会显示系统当前的星期数并显示中文星期的格式。如图 13-5 所示。

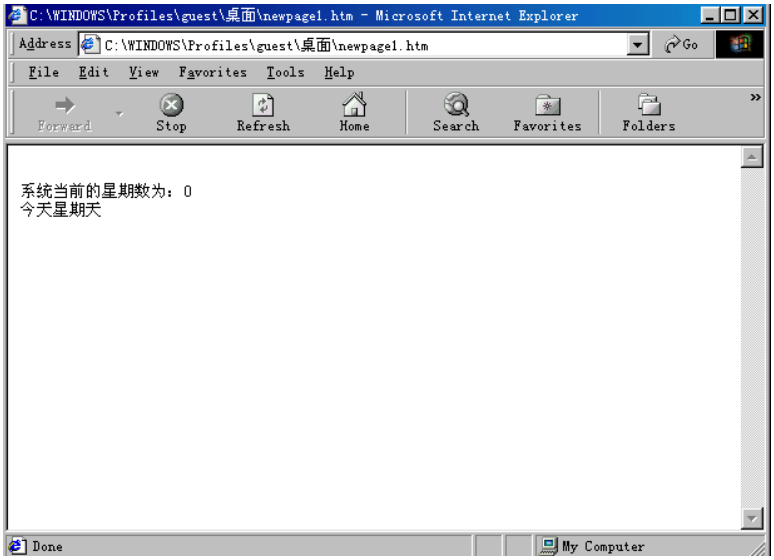


图13-52

13.37.6 getTime

该方法返回一个代表当前日期的整数值。这个整数值为从 1970 年 1 月 1 日 0 时算起的毫秒数。在页面的设计或是脚本设计的过程中，常利用该方法来比较两个日期对象值的大小。

该方法的调用格式如下：

dateobj.getTime()

请看下边的实例：

```
<html>

<head>
<title>
</title>
</head>

<body>

<p>
<script language="JavaScript">
<!--
thisday=new Datez()
time=thisday.getTime()
document.write("<br>系统当前的时间数为："+time)
// -->
</script>
</p>
</body>
</html>
```

在页面中调用该脚本就会返回一个很大的数值：972203415810，大家算一下是不是和1970 年起的毫秒数相符合。如图 13-6 所示。

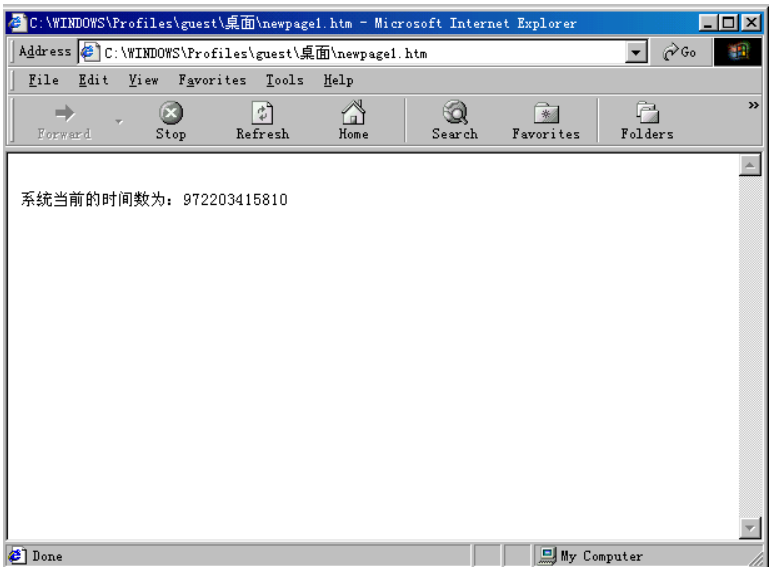


图13-53

对于获取系统设置的日期对象的时间，JavaScript 中还设置了如下的几个方法，分别表示为：

- **getHours**：该方法获得日期对象的小时数;
- **getMinutes**：该方法获得日期对象的分钟数;
- **getSeconds**：该方法获得日期对象的秒数。

在以上的设置中，系统返回的都是以 0 为起始的数字。其中在设置小时数的过程中，设置的小时数为 24 小时制的格式。

这些方法的调用格式同上面格式的应用是一样的。在下面的实例中我们来看看其具体的设置情况。

```
<html>

<head>
<title>
</title>
</head>

<body>

<p>
<script language="JavaScript">
<!--
thisday=new Date()
time=thisday.getHours()
document.write("<br>系统当前的时间数为："+time)
min=thisday.getMinutes()
document.write("<br>系统当前的时间数为："+min)
```

```
sec=thisday.getSeconds()
document.write("<br>系统当前的时间数为："+sec)

// -->
</script>
</p>
</body>
</html>
```

在浏览器中浏览该页面的时候，就会显示系统日期的时间数，用户在调用的过程中，不妨和系统的时间比较一下。如图 13-7 所示。

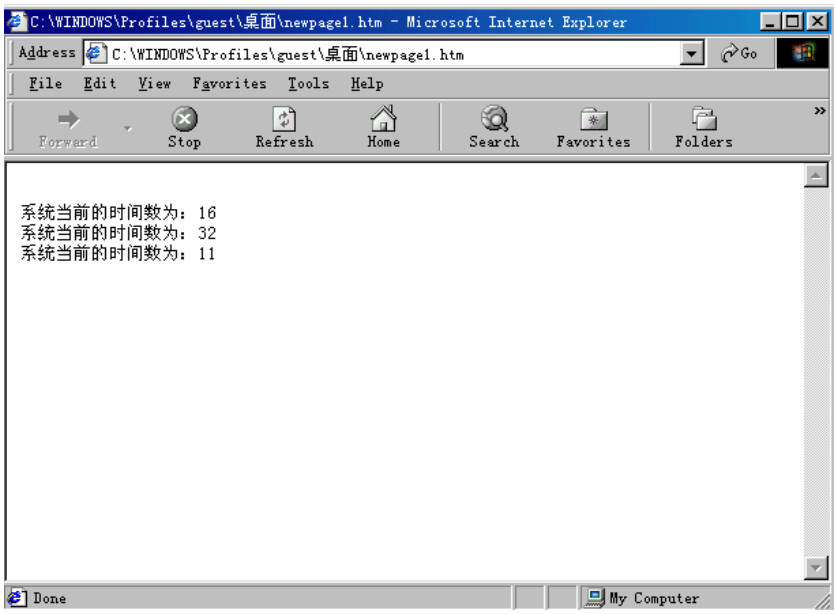


图13-54

13.37.7 setTime

根据编程的经验，带有 set 的函数一般是用来设置函数值的。事实上在 date 对象中，系统设置了许多的 set 函数，以便来设置系统的日期。该 setTime 函数用来设置系统的时间值。在该方法的应用中，它返回一个自 1970 年 1 月 1 日零时的毫秒数。

该方法的应用格式如下：

dateobj.setTime(arg)

其中：

- **dateobj**：设置的时间对象；
- **arg**：设置的参数。

在页面设计中，该实例的应用如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>New Page 1</title>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
```

```
</head>
<body>
<p>
<script language="JavaScript">
<!--
setdy=new Date("08/19/1999")
res=setdy.getTime()
day=new Date()
rs=day.setTime(res)
alert("设置时间："+rs+"毫秒")
// -->
</script>
</p>
</body>
</html>
```

在该脚本中，先设置一个时间对象，该对象的值为 1999 年 8 月 19 日，然后再利用 getTime 方法取得该时间与系统起始时间的毫秒数的比较值，将该值赋给设置的变量，然后再设置一个时间变量，然后调用 window 对象的 alert()方法，显示该时间数。

其中：

- **setdy=new Date("08/19/1999")**：设置一个时间对象；
- **res=setdy.getTime()**：调用时间对象的 gettime 方法，设置毫秒数，然后将该数字赋给设置的变量 res；
- **day=new Date()**：设置系统当前的时间；
- **rs=day.setTime(res)**：设置毫秒数；
- **alert("设置时间："+rs+"毫秒")**：调用 window 的对象的 alert()方法，在文档上显示用户的设置。

该页面显示情况如图 13-8 所示。

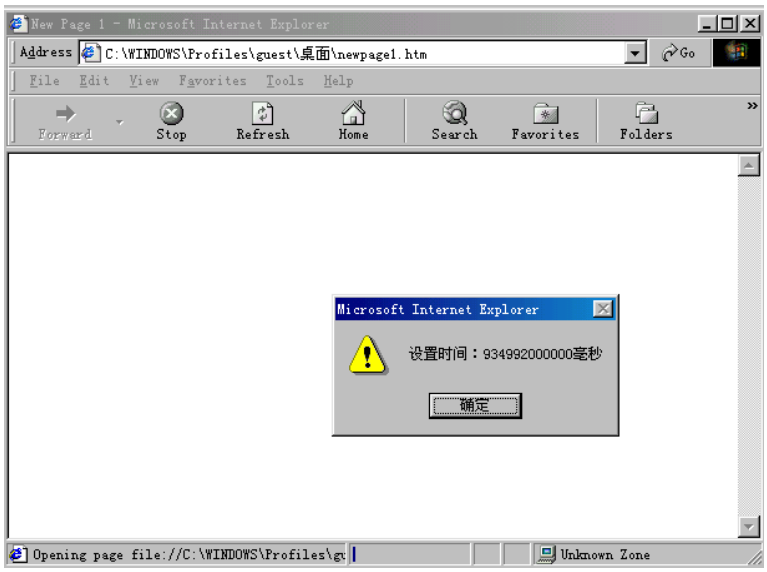


图13-55

13.37.8 setYear

该方法用来设置指定的年份。在该方法的应用中，需要设置一个两位数的年份来表示与 1900 年的差值。

该方法的应用格式与 setTime 方法一样。

请看下边的实例：

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>New Page 1</title>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
</head>

<body>

<p>
<script language="JavaScript">
<!--
setdy=new Date("08/19/1999")
res=setdy.getYear()
day=new Date()
rs=day.setYear(res)
if(confirm("设置时间年份为："+res+"年"))
document.write("用户设置的年份为："+(1900+res)+"年。")
// -->
</script>
</p>
</body>
</html>
```

该脚本是在上面脚本的基础上改编而成的。在该实例中，先设置一个时间对象，然后取得该时间对象的年份值，再将该时间对象值赋给设置的变量，然后在利用本例的函数 setYear 方法，在文档中显示用户设置的年份。在显示的过程中，设置一个对话框，如果用户确认输入的数据就在文档中显示用户的设置。

其中：

- **setdy=new Date("08/19/1999")**：设置时间对象;
- **res=setdy.getYear()**：取得时间对象的年份值;
- **day=new Date()**：设置系统当前时间。设置时间对象;
- **rs=day.setYear(res)**：设置年份;
- **if(confirm("设置时间年份为："+res+"年"))**：设置一个确认对话框，并设置了一个判断对象。如果用户按下确定对话框，则执行下边的语句;
- **document.write("用户设置的年份为："+(1900+res)+"年")**：在文档中显示用户设置的年份。

在浏览器中，该脚本的执行情况如图 13-9 所示。当用户按下【确定】按钮后，在窗口中显示的信息如图 13-10 所示。

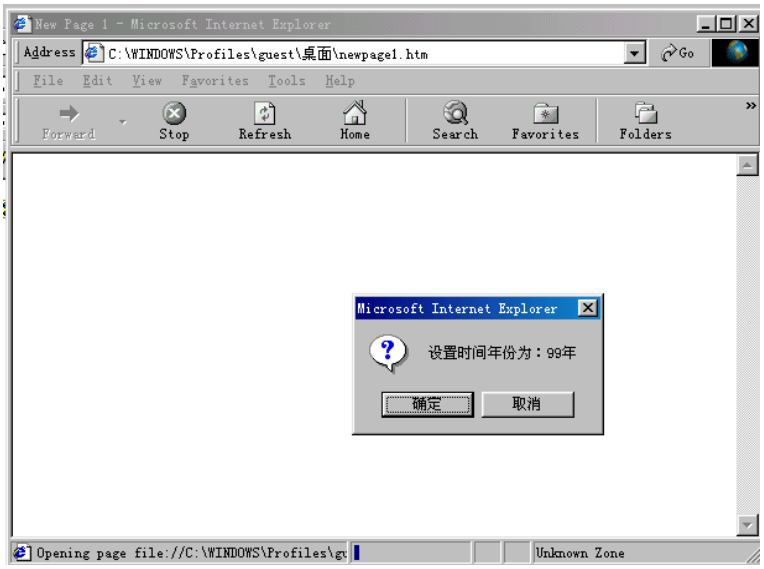


图13-56

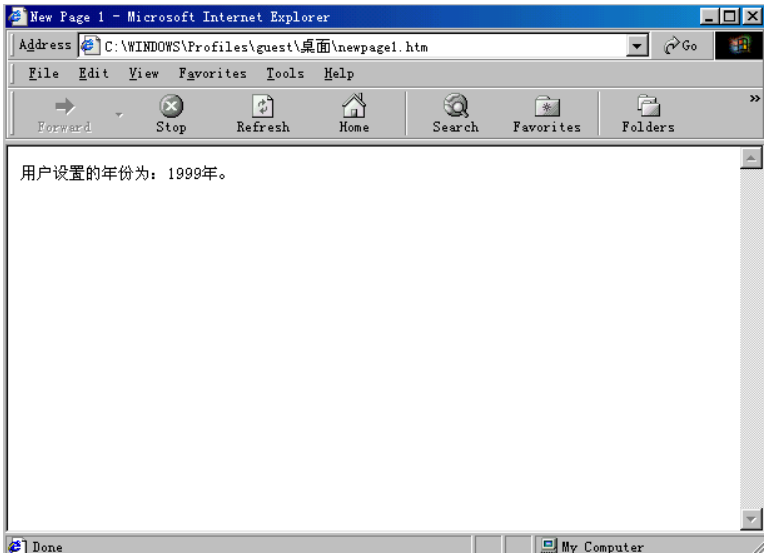


图13-57

与该方法一样，在 JavaScript 中，与其功能一样的方法还有：

- **setMonth()**：设置日期对象的月份；
- **setDate()**：设置日期；
- **setDay()**：设置星期；
- **setHours()**：设置小时数；
- **setMinutes()**：设置分钟数；
- **setSeconds()**：设置秒数。

这些方法的调用格式都是一样的，格式如下：

dateobj.method(arg).

读者可以试着将上面的例子改一下，看看程序运行的效果。

13.37.9 getTimezoneOffset

该方法以格林尼治时间为标准，返回一个分钟为单位的差值。一般来说该值为一个负数。该方法的调用格式如下：

dateobj.getTimezoneOffset()

在页面中，其效果表现如图 13-11 所示。

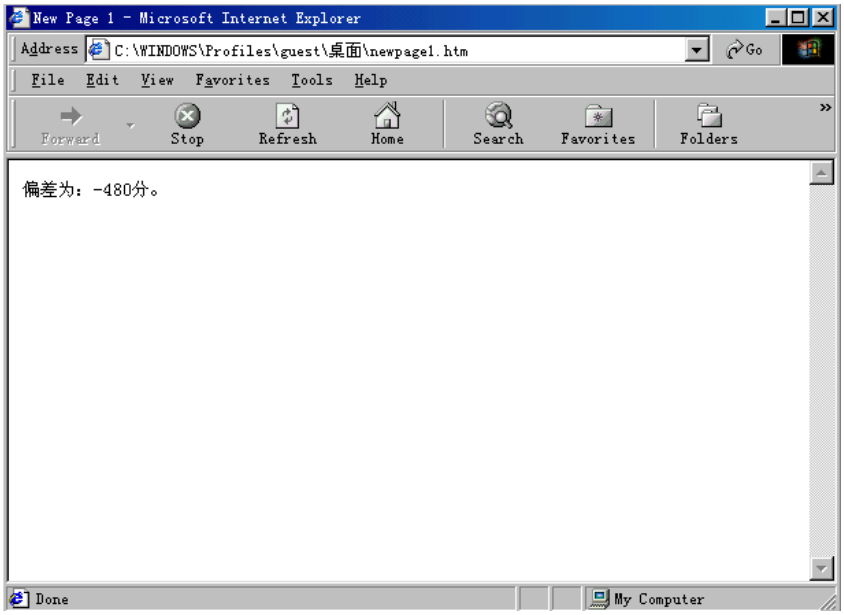


图13-58

源代码如下：

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>New Page 1</title>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
</head>

<body>

<p><script language="JavaScript">
<!--
day=new Date()
rs=day.getTimezoneOffset()
document.write("偏差为："+rs+"分。")
// --></script></p>
</body>
```

```
</html>
```

13.37.10 toGMTString（）

将一个日期对象转换为一个字符串。这是按照 internet 格式设置的对象字符串的值。在时间的对象转换上，也许随着系统操作平台的不同而有所变换，但一般的字符串格式为“星期、月、日、年、时、分、秒”。

该方法的使用格式如下：

dateobj.toGMTString()

应用示例如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>New Page 1</title>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
</head>
<body>
<p><script language="JavaScript">
<!--
day=new Date()
rs=day.toGMTString()
document.write("时间为："+rs)
// --></script></p>
</body>
</html>
```

在浏览器中，该脚本显示如图 13-12 所示。

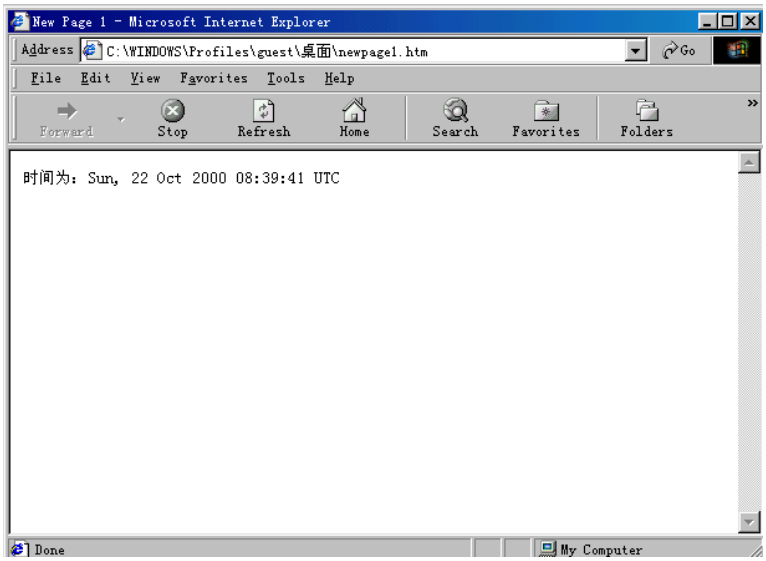


图13-59

13.37.11 toLocaletring（）

该方法将日期对象转换为本地的日期格式，恰好与上面的方法相对。显示的格式依赖于系统的平台。

该方法的利用格式如下：

dateobj.toLocaletring()

将上面的例子改一下，显示的效果如图 13-13 所示。

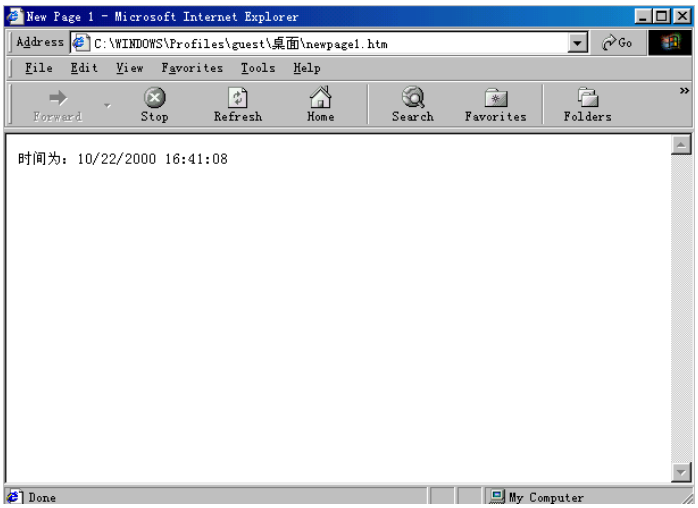


图13-60

源代码为：

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>New Page 1</title>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
</head>

<body>

<p><script language="JavaScript">
<!--
day=new Date()
rs=day.toLocaleString()
document.write("时间为："+rs)
// --></script></p>
</body>
</html>
```

在后续章节的例子中，将讲述时间对象的格式化问题。感兴趣的读者可以先去看看相应的章节。

第14章 数学对象

主要内容

- 数学对象的属性
- 数学对象的方法

本章导读

数学是一切学科的基础学科，所以对于所有的程序设计语言来说都规定了一些关于数学运算的语句。JavaScript 也不例外。在 JavaScript 中，math 对象就提供了一些数学函数和常数代码。和 string 对象一样，该对象属于 JavaScript 的内建对象。这里引入的内建对象指的是标准的数据定义语言，受大多数浏览器支持，受不同版本的语言支持，用户可以直接引用，而无需考虑对象的情况。在 math 对象中，JavaScript 仍然将它们分为属性和方法来讲解，下面来看看这些属性和方法。

14.38 math 对象的属性

math 对象的属性就是一些常用数学常数，只是这里将它作为属性来讲解罢了。

14.38.1 E

该属性为欧拉常数，是自然对数的底数，常用在数学运算中。这个数的值一般为 2.71828。

该函数的引用格式如下：

math.E

在页面设计过程中，该属性的应用情况如下：

```
<html>

<body>

<p>
<script language="JavaScript">
<!--
a=Math.E
alert("math.E="+a)
// -->
</script>
</p>
</body>
</html>
```

其中：

- **a=Math.E**：设置一个变量，取得用户设置的 math 对象的值；
- **alert("math.E="+a)**：调用 window 对象的 alert（）方法，将用户设置的值显示在页面上。这样就设置了该属性的值。

在浏览器中，该页面的表现情况如图 14-1 所示。

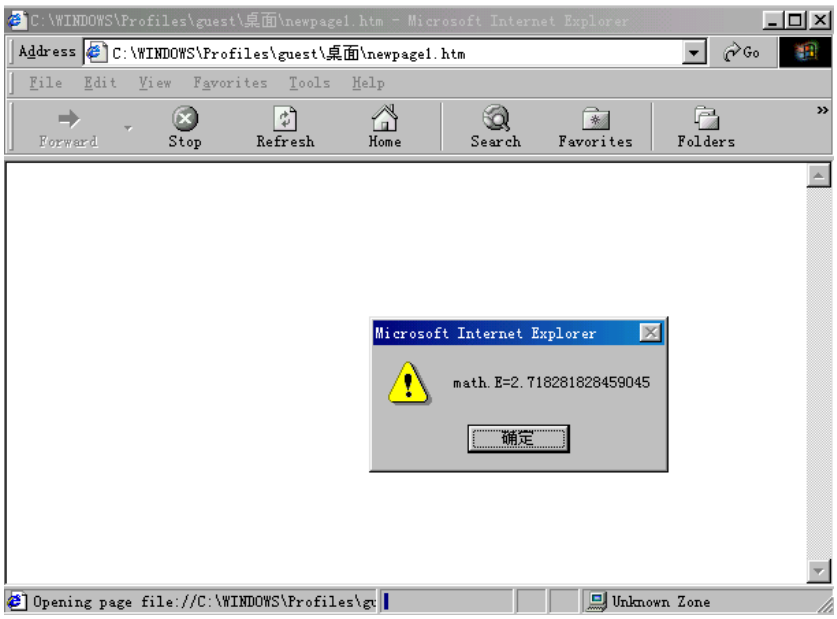


图14-61

14.38.2 其 它 属 性

在 JavaScript 中，math 对象还有其他的属性，下面分别列出，并加以简短地说明。事实上，它的应用同上面的一样，设置的方法也是一样的。

- **LN2**：2 的自然对数；
- **LN10**：10 的自然对数；
- **LOG2E**：底数为 2，真数为 E 的对数；
- **LOG10E**：底数为 10，真数为 E 的对数；
- **PI**：圆周率的值；
- **SQRT1_2**：0.5 的平方根；
- **SQRT2**：2 的平方根。

在页面的设置情况如下：

```
<html>

<head>
<title>C:\Apache\htdocs\MATH.E.htm</title>
</head>

<body>

<p><script language="JavaScript">
<!--
a=Math.LN2
document.write("<br>math.LN2="+a)
b=Math.LN10
```

```
document.write("<br>math.LN10="+b)
c=Math.LOG2E
document.write("<br>math.LLOG2E="+c)
d=Math.LOG10E
document.write("<br>math.LLOG10E="+d)
f=Math.PI
document.write("<br>math.PI="+f)
g=Math.SQRT1_2
document.write("<br>math.SQRT1-2="+g)
h=Math.SQRT2
document.write("<br>math.SQRT2="+h)
// -->
</script> </p>
</body>
</html>
```

其中：

- **a=Math.LN2**：设置数学对象的 LN2 属性，并将该值赋给设置的变量 a；
- **document.write("
math.LN2="+a)**：在文档中显示设置的值；
- **b=Math.LN10**：设置数学对象的 LN10 属性，并将该值赋给设置的变量 b；

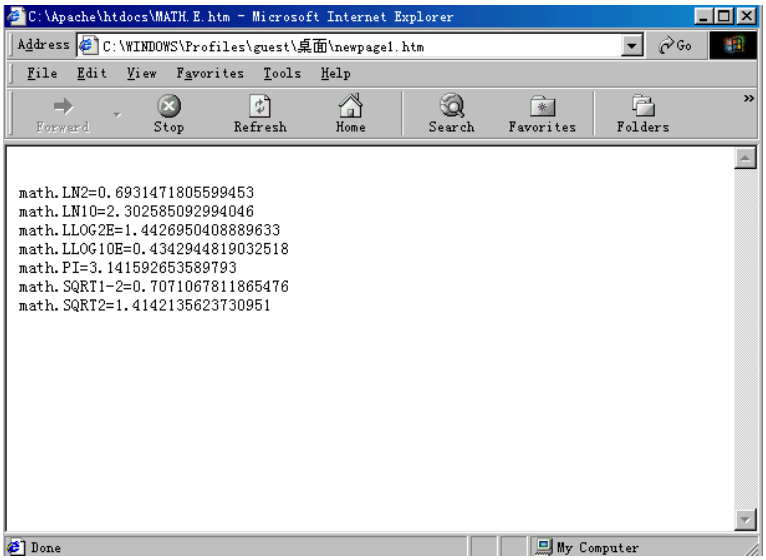


图14-62

- **c=Math.LOG2E**：设置数学对象的 Log2E 属性，并将该值赋给设置的变量 c；
- **d=Math.LOG10E**：设置数学对象的 LOG10E 属性，并将该值赋给设置的变量 d；
- **f=Math.PI**：设置数学对象的 PI 属性，并将该值赋给设置的变量 f；
- **g=Math.SQRT1_2**：设置数学对象的 SQRT1-2 属性，并将该值赋给设置的变量 g；
- **h=Math.SQRT2**：设置数学对象的 SQRT2 属性，并将该值赋给设置的变量 h。

在浏览器中，该页面的显示情况如图 14-2 所示。

14.39 math 对象的方法

数学对象的方法是常用的函数库，在设置页面的过程中可以直接引用系统设置的函数——属性对象的方法。

14.39.1 abs

该方法求用户设置数的绝对值。在利用该方法的时候，如果用户设置的数为一个负数，那么就可以利用该方法将该数值转换为相应的正数形式。

该方法的调用方式如下：

Math.abs(arg)

其中：

- **Math**：是 JavaScript 内建的对象，该值为一个关键字，在使用的过程中，必须跟在方法的前面；
- **Arg**：设置的参数。

在页面的设置过程中，调用的情况如下：

```
<html>

<head>
<title></title>
</head>

<body>

<p>a=-5</p>

<p><script language="JavaScript">
<!--
a=-5
b=Math.abs(a)
document.write("a 的绝对值为："+b)
// -->
</script>
</p>
</body>
</html>
```

其中：

- ** a=-5 **：在页面中设置一个数；
- **a=-5**：设置参数；
- **b=Math.abs(a)**：调用数学对象的 abs 方法，处理用户设置的参数，然后将取得值赋给设置的变量 b；
- **document.write("a 的绝对值为："+b)**：在页面中显示设置的变量值。

在浏览器中，该页面的表现情况如图 14-3 所示：

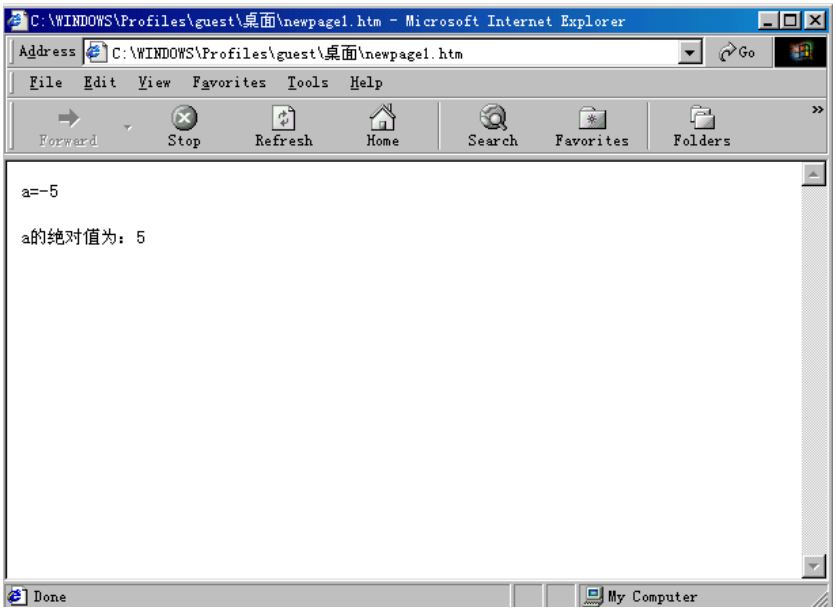


图14-63

14.39.2 acos

该方法求用户给定值的反余弦函数的值。在使用该函数的时候要注意，在给定的变量中，变量的有效范围应该是（-1，1），如果超出这个范围，那么在脚本的调试中，就会返回一个 0 或 Nan 值。用户利用该方法时一定要注意。

该方法的应用格式如下:

Math.acos（arg）

其中：

- **arg**：指用户设置的参数。1-〈arg〈1。

在页面设计的格式如下：

```
<html>

<head>
<title>math</title>
</head>

<body>

<p><script language="JavaScript">
<!--
a=1
c=2
b=Math.acos(a)
confirm("acos(1)="+b)
document.write(b=Math.acos(c))
// -->
```

```
</script> </p>
</body>
</html>
```

其中：

- **a=1**：定义一个变量，变量的值为 1；
- **b=Math.acos(a)**：设置变量 b，该变量的值就是用户对变量 a 求的余弦值；
- **confirm("acos(1)="+b)**：调用 window 对象的 confirm()方法，让用户确认求得的值；
- **document.write(b=Math.acos(c))**：这里设置变量 c 的值为 2，就是看看当变量的值超过系统设置的范围时，函数返回的值。该语句将该值在文档中显示出来。

在浏览器中，脚本执行的情况如图 14-4 所示。

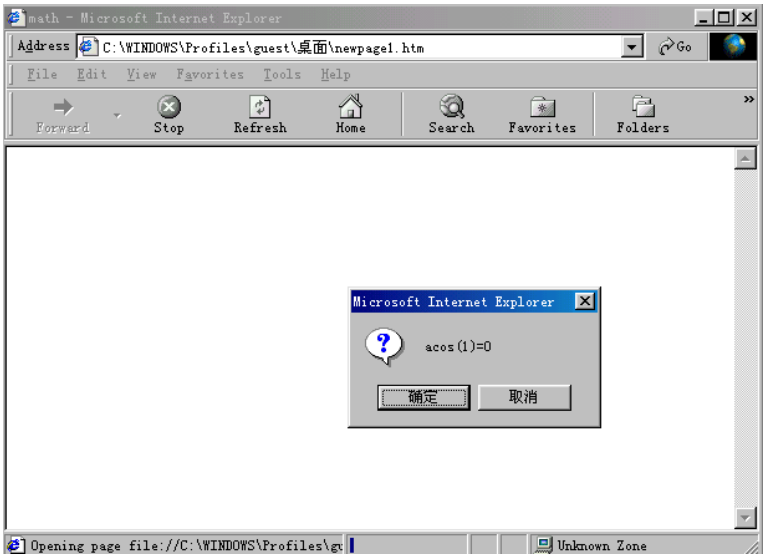


图14-64

对于上面的函数，我们称之为三角函数，在 JavaScript 中，一共提供了 7 个三角函数，除了上面讲的 acos 而外，还有如下几个：

- **cos()**：计算变量的余弦值；
- **sin()**：计算变量的正弦值；
- **tan()**：计算变量的正切值；
- **asin()**：计算变量的反正弦函数值；
- **atan()**：计算变量的反正切值；
- **atan2**：计算变量的反余切值。

下面来看看在脚本设计中函数的执行情况：

```
<html>
<head>
<title><math/title>
</head>
```

```
<body>

<p>
script language="JavaScript">
<!--
a=1
with(Math){
b=cos(a)
c=sin(a)
d=tan(a)
e=asin(a)
f=atan(a)
g=atan2(a)
}
document.write("当 a=1 时<br>")
document.write("<br>cos(a)="+b)
document.write("<br>sin(a)="+c)
document.write("<br>tan(a)="+d)
document.write("<br>atan(a)="+f)
document.write("<br>atan2(a,0)="+g)
document.write("<br>asin(a)="+e)
// -->
</script>
</p>
</body>
</html>
```

在该脚本的设置中，我们先设置变量 a=1，然后再利用 with 语句，设置脚本对象为 math 对象，然后分别设置三角函数的值，最后调用文档对象的 write 方法在文档窗口中将求的结果在文档中显示出来。

在浏览器中，该页面的显示情况如图 14-5 所示。

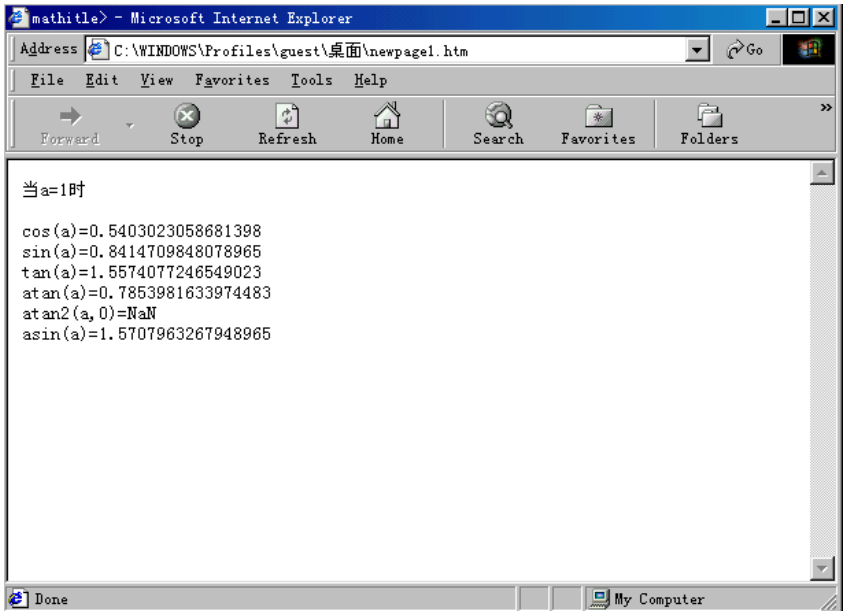


图14-65

14.39.3 max

该方法为一个大小比较函数，在设计脚本的过程中，设置两个参数，然后通过该方法的设置，就会返回两个函数中较大的数。

该方法的调用格式如下:

math.max(arg1,arg2)

其中：

- **arg**：表示用户设置的参数。

该方法在页面脚本中设计的应用格式如下：

```
<html>

<head>
<title></title>
</head>

<body>
<script language="JavaScript">
<!--
arg1=12
arg2=13
arg3=10

a=Math.max(12,13)
b=Math.max(arg2,arg3)

document.write("12 与 13 中较大的为："+a)
document.write("<br>13 与 10 中较大的为："+b)
/ --></script>
</body>
</html>
```

其中：

- **arg**：系统设置的参数；
- **a=Math.max(12,13)**：设置变量，将用户比较所得到的值赋给设置的变量。该语句获得两个数并比较获得的最大值；
- **document.write("12 与 13 中较大的为："+a)**：在文档中显示用户比较得到的值。

在浏览器中，脚本运行的情况如图 14-6 所示。

与 max 方法相对，求最小值函数的方法为 min()，该方法的使用与设置方法同 max 一样，同样设置两个变量，通过该方法的设置，获得其中最小的一个值。该方法的调用格式如下：

Math.min(arg1,arg2)

对于以上的两个函数，一般称之为比较函数。其中，min 方法获得用户设置的最小值，而 max 方法获得用户设置的最大值。可以利用这两个函数来进行排序的操作。

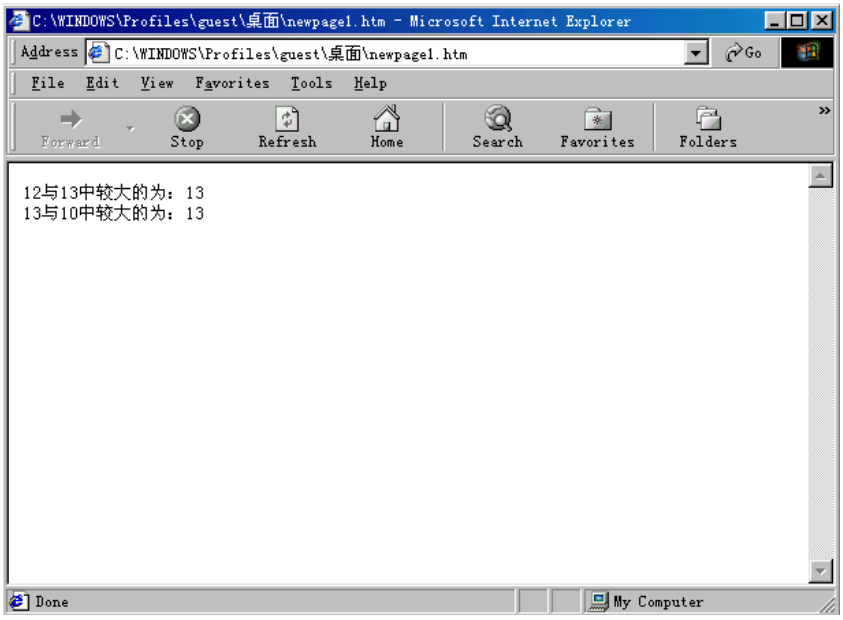


图14-66

14.39.4 round

该方法将一个数值，特别是浮点数舍入成它最近的一个整数。在设置的过程中，一般给定一个浮点数参数变量，然后再利用该方法将变量求值，然后再进行四舍五入的操作。即如果小数部分的值大于 0.5，整数部分加 1，否则，就舍掉小数部分的数值。

该方法的应用格式如下：

Math.round(arg1)

其中：

- **arg1**：为一个浮点数的变量。如果该数已经为一个整数，利用该方法就失去意义了。

在页面设置中，该方法的利用格式如下：

```
<html>

<head>
<title></title>
</head>

<body>
<script language="JavaScript">
<!--
arg1=12
arg2=16
```

```
arg3=10

a=Math.round(12/13)
b=Math.round(arg2/arg3)

document.write(a)
document.write("<br>"+b)

/ --></script>

</body>
</html>
```

在该脚本的设置中，我们设置两三个变量，然后进行两两相除的运算，再调用 round() 方法，将求的值四舍五入，最后在页面中显示出来。在浏览器中浏览该页面，页面的显示情况如图 14-7 所示。

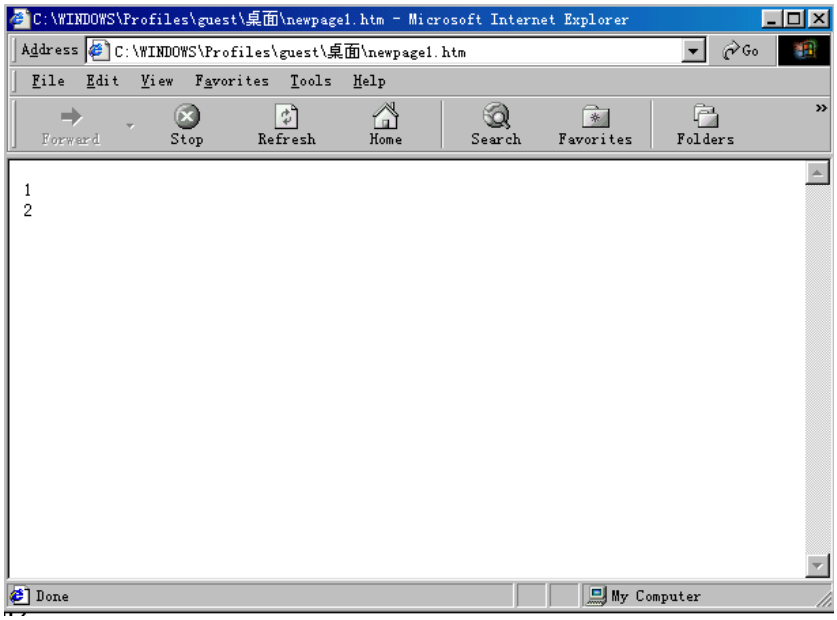


图14-67

一般的，我们将上面的 round（）函数称之为舍入函数，同该函数相对的函数还有如下的两个，分别为：

- **floor**：该方法与 round 相反，求的是小于或等于变量的值；
- **ceil**：该方法求的是大于或等于变量的值。

这些方法与 round 方法归属于一类，其方法的设置和调用是一致的。请看下边的实例。

```
<html>

<head>
<title></title>
</head>

<body>
```

```
<script language="JavaScript">
<!--
arg1=16
arg2=10
a=arg1/arg2
b=Math.round(a)
c=Math.floor(a)
d=Math.ceil(a)
document.write(a)
document.write("<br>"+b)
document.write("<br>"+c)
document.write("<br>"+d)
/ --></script>

</body>
</html>
```

其中：

- **arg1=16**：设置变量 1；
- **arg2=10**：设置变量 2；
- **a=arg1/arg2**：设置变量 a 的值为两个变量相除的值；
- **b=Math.round(a)**：设置 round 方法；
- **c=Math.floor(a)**：设置 floor 方法；
- **d=Math.ceil(a)**：设置 ceil 方法；
- **document.write(a)**：在文档中将设置的变量的值显示出来。

在页面中，其显示情况如图 14-8 所示。

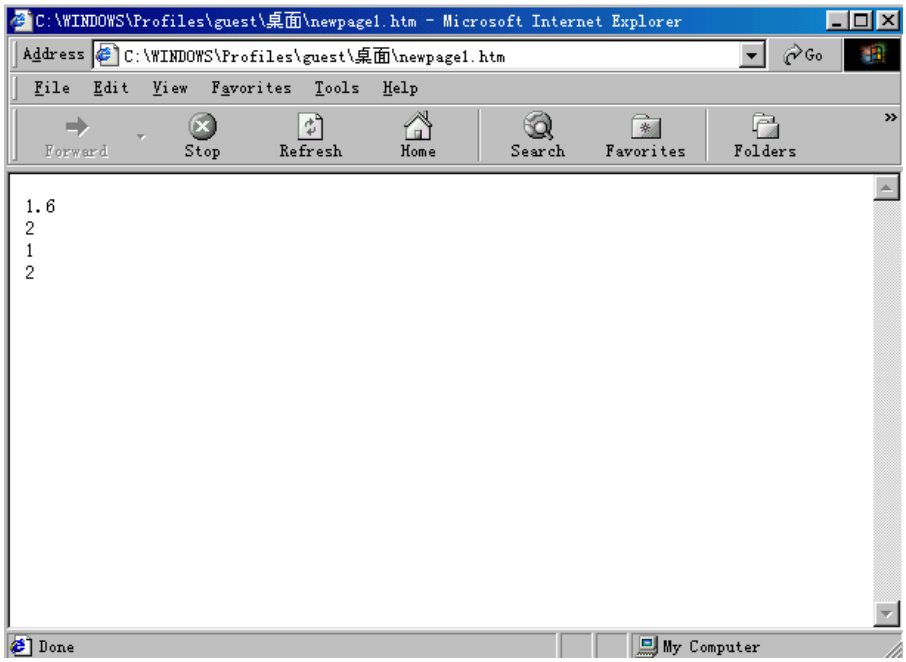


图14-68

14.39.5 random

该方法产生一个位于 0 和 1 之间的随机数。该方法在一些测验设置中是相当有效的，系统可以调用该函数方法，随机地产生试题，然后让用户作答，也可以设置随机的效果，显示页面的欢迎词等等信息。

该方法的调用格式如下；

Math.random()

在页面设计中，脚本的调用情况如下：

```
<html>

<head>
<title></title>
</head>

<body>
<script language="JavaScript">
<!--

b=Math.random()
c=Math.random()
document.write("<br>"+b)
document.write("<br>"+c)

/ --></script>

</body>
</html>
```

其中：

- **b=Math.random()**：设置随机效果，该语句产生一个为 0 到 1 之间的随机数，并将该数值赋给设置的变量 b；
- **document.write("
"+b)**：在文档中写出该结果。

在浏览器中，该页面的表现情况如图 14-9 所示。

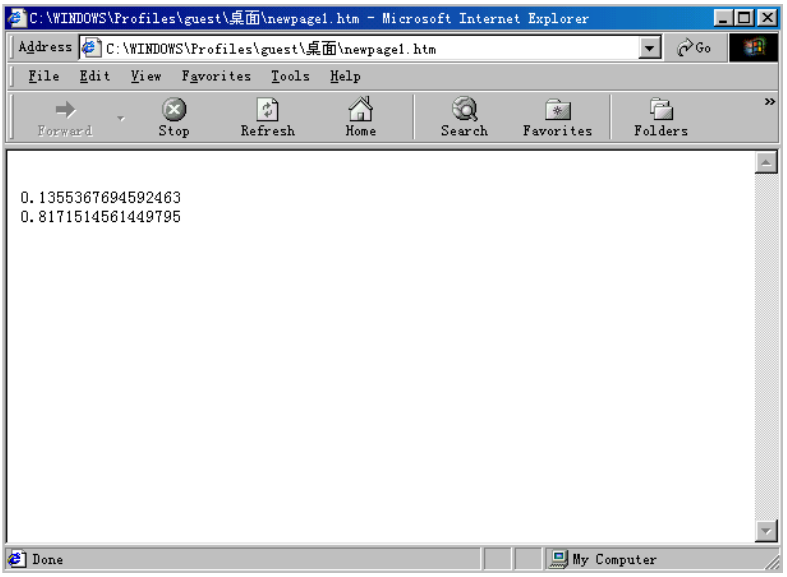


图14-69

在 math 对象中，还存在一些三角函数方法，在下面的例子中，给出了相应的具体设置方法。

```
<html>

<head>
<title></title>
</head>

<body>
<script language="JavaScript">
<!--
a=2
b=3
with(Math){
d=exp(a);
c=log(a);
e=pow(a,b);
f=sqrt(b)
}
document.write("<br>"+c)
document.write("<br>"+d)
document.write("<br>"+e)
document.write("<br>"+f)
/ --></script>

</body>
</html>
```

其中：

- **a=2**：设置变量 a；
- **b=3**：设置变量 b；

- **with(Math)**：设置 math 对象捷径；
- **d=exp(a)**：该方法返回一个指数函数值；
- **c=log(a)**：该方法返回一个以 e 为底的对数值；
- **e=pow(a,b)**：求幂方法；
- **f=sqrt(b)**：求平方根函数；
- **document.write("
"+c)**：在文档中写入函数值。

对于以上的函数，系统将他们归类为三角函数，通过这些方法的设置，可以完成一些常用的数学计算。在页面中，该脚本的执行情况如图 14-10 所示。

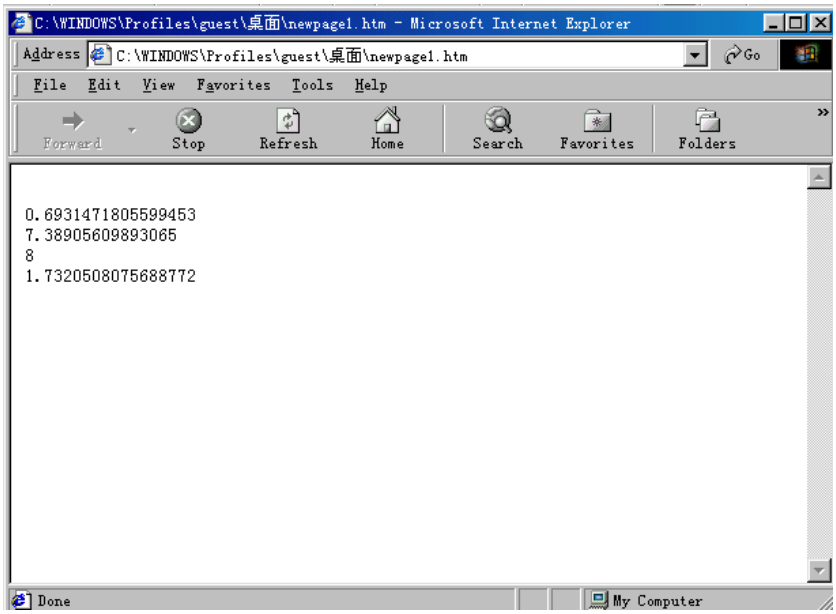


图14-70

第15章 数组对象

主要内容

- 数组对象的属性
- 数组对象的方法

本章导读

JavaScript 中，数组对象是提供给用户进行存储与处理有序数据集合的数据结构。事实上，在前面的对象中，我们已经接触了大部分数组的对象，像表单集、帧组等等。在后续的章节中，设置的有关用户的查询、cookie 数组的实现，都要使用到本章讲到的数组对象。

细心的用户或是接触过 c 语言的用户都有这样的感觉，JavaScript 中语言的设置和 c 语言中设置函数的方法差不多。在数组的设置中，两者都有异曲同工之妙。

在 JavaScript 脚本语言中，将数组设置为一个新的对象。对数组对象的操作就是操作用户定义的新对象。

在如下的章节中，我们将学习数组对象的设置和操作。

15.40 数组对象的创建

数组对象创建的格式如下：

```
function students(name,age)
{this.name=name;
this.age=age;
}
```

好了，我们创建了一个数组对象，下面利用一个 new 关键字来引用该变量，建立一个实例。

stu1=new students("黎明",28)。

这里建立的实例中，设置的新对象中的属性为 name 与 age 两项。在建立的实例中，设置的一个对象为 stu1，其中 name 属性值为“黎明”，age 属性值为“28”。在实例的引用中，格式如下：

```
stu1.name
stu1.age
```

这样我们就完成了一个实例的定义，在页面中显示设置的对象属性值的源代码如下：

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>New Page 1</title>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
</head>

<body>

<p><script language="JavaScript"><!--

function stu(name,age)
{ this.name=name;
this.age=age;
}
stu1=new stu("黎明",28)
document.write(stu1.age)
// -->
```

```
</script>
</p>
</body>
</html>
```

其中：

- **document.write(stu1.age)**：在页面中显示设置对象的属性值。这里为对象 stu1 年龄值。

在浏览器中。该页面显示的效果如图 15-1 所示。

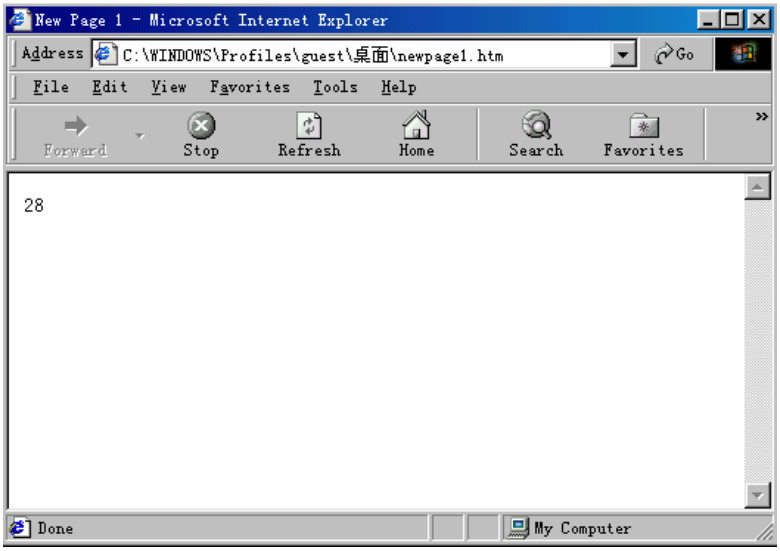


图15-71

15.41 数组对象的扩充

在上面的例子中，数组对象还可以进一步地扩充，我们可以引入另一个数组对象的属性，从而构成数组对象间的引用。

我们可以进一步定义一个数组对象，设置上面对象的源为一所学校，该对象为源对象的子类。

如：

```
function school(sname,stu)
{this.sname=sname;
this.stu=stu;
}
```

定义了一个新的数组对象之后，再来引用它，建立一个实例，过程如下；

```
sstu=new school("四川大学",stu1)。
```

通过这样的引用，stu1 的各种属性值就被 sstu 所应用。在调用 stu1 设置的属性格式如下：

- **sstu.stu.name**：调用设置的 name 属性；
- **sstu.stu1age**：调用设置的 age 属性。

综合以上的设置，在页面中，该脚本源代码如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>New Page 1</title>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
</head>
<body>
<p><script language="JavaScript"><!--
function school(sname,stu)
{
this.sname=sname;
this.stu=stu;
}
function stu(name,age)
{ this.name=name;
this.age=age;
}
stu1=new stu("黎明",28);
sstu=new school("四川大学",stu1);
document.write(sstu.stu.name) ;
// -->
</script> </p>
</body>
</html>
```

在浏览器中，该页面的显示情况如图 15-2 所示。

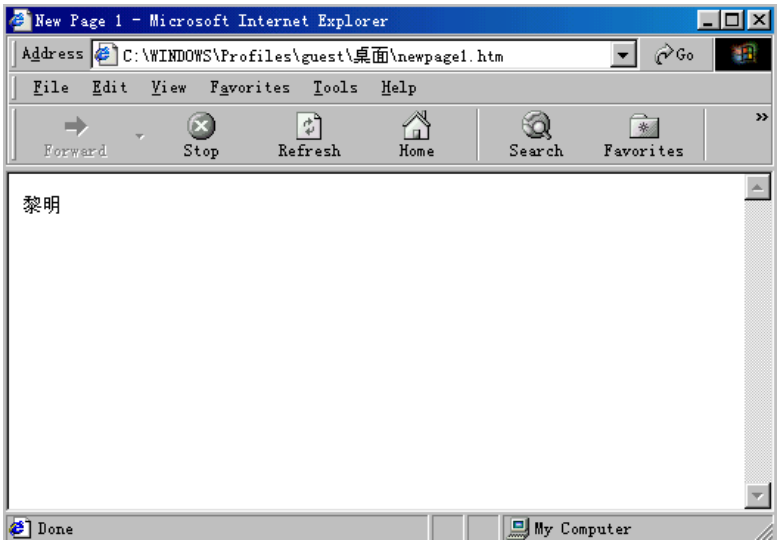


图15-72

15.42 对象类数组

以上讲述了数组对象的设置和数组对象属性的引用问题。下面来看看对象类数组对象的设置。

在对象类数组对象的设置过程中，JavaScript 中设置了关键字 new Array 来实现。例如，假设现在设置学生姓名数据，定义一个长度为 5 的数组。该定义的格式如下：

stu= new Array(5)z

在实际的输入实例中，应用的格式如下：

stu[0]="黎明"

stu[1]="立志"

stu[2]="红鱼"

stu[3]="赵站"

stu[4]="张云"

从上例可以看出，数组对象的定义都是从关键字的定义开始的，然后利用数组对象输入相应的值。在数组对象的下标中，都是从 0 开始记数的。

对象的调用格式同上面介绍的一样，比如在页面显示设置的变量，则格式如下：

document.write(stu[0])

将上面的语句组织成脚本的源代码如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>New Page 1</title>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
</head>
<body>
<p>
<script language="JavaScript"><!--
stu= new Array(5)
stu[0]="黎明"
stu[1]="立志"
stu[2]="红鱼"
stu[3]="赵站"
stu[4]="张云"
document.write(stu[0]) ;
// -->
</script>
```

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

在浏览器中，显示的情况如图 15-3 所示。

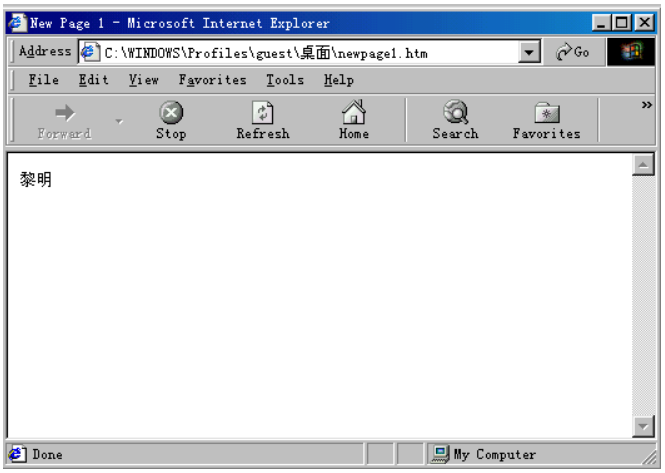


图15-73

下面来看看数组对象的属性。在 JavaScript 中，常使用到的属性为 length。该属性用来获得数组对象的长度。该属性的调用格式如下：

array.length

其中：

- **array**：设置的数组对象。

请看下边的实例：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>New Page 1</title>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
</head>
<body>
<p><script language="JavaScript"><!--
stu= new Array()
stu[0]="黎明"
stu[1]="立志"
stu[2]="红鱼"
stu[3]="赵站"
stu[4]="张云"
document.write(stu.length) ;
// -->
</script>
</p>
</body>
</html>
```

在该脚本的设置中，设置了一个数组，该数组的长度未知，然后设置了五个实例，最

后利用数组对象的 length 属性获得数组对象的实例值，并在页面中显示数组对象的长度。

在浏览器中，该页面的显示情况如图 15-4 所示。

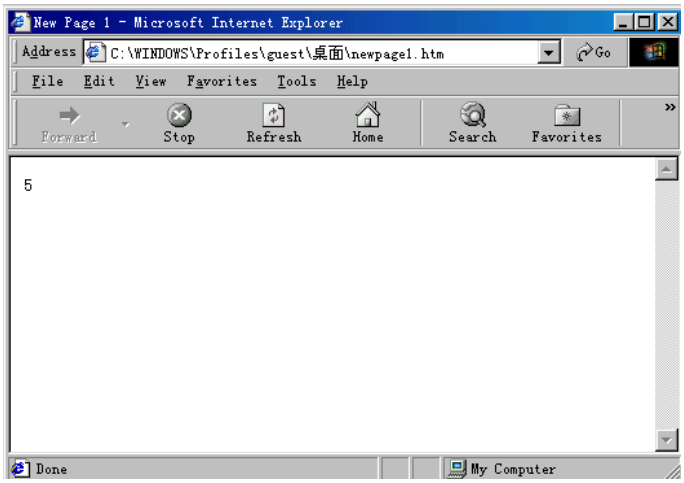


图15-74

第16章 样式单实例

主要内容

- 样式单入门
- 样式单的定义

本章导读

利用样式单可以设置布局统一的页面：对于商业站点，可以提高企业页面的风格和企业文化的内涵；对于个人网页，有助于个性化页面的实现。本章主要通过具体的实例，介绍了样式单的设置以及利用样式单来控制页面的显示。

16.43 样式单的实用

在 html3.2 语言中，指定了格式语言的显示和排版规定。这样，我们可以定义格式页面的设置，以便于方便和快速地维护文档页面，这样定义的格式样式单，我们称之为级联样式单。

样式单提供了对控制标题、段落、清单、HTML 元素布局和显示的功能。这样，web 网页设计人员可以不断设计出自己想要的格式。有了样式单，可以指定各级标题的颜色、字体的大小，可以精确地设置背景的色彩、背景图案所在的区域、重叠方式，可以指定边框的大小，外观等等。同时，可以方便地利用外界的样式模式，可以改变文本的样式。总之，利用样式单可以设置布局统一的页面，对于商业站点，可以提高企业页面的风格和企业文化的内涵，对于个人网页，有助于个性化页面的实现。

下面，将以一个实例说明样式单的作用。实例的源代码如下：

```
<html>

<head>
  <style type="text/css">
  <!—
    h1{
      color: green;
      font-style: italic;
      font-size: 12pt;
    }
    h1.special{
      font-size: 24pt;
    }
    .newheader{
      color: yellow;
    }
    #caps{
      font-variant: small-caps;
    }
  -->
  </style>
</head>
<body>
  <h1>Green, italic, and a point size of 12</h1>
  <h1 class="newheader">Yellow, italic, and a point size of 12</h1>
  <h1 class="special">Point size of 24</h1>
  <p>
    Here is some <span id="caps">capitalized</span> text.
  </p>
</body>
</html>
```

其中：

- **<style type="text/css">：设置样式单的定义**

样式单在程序的源代码中，都是以这种格式开头的，在页面的执行过程中，通过这样的表头，告诉浏览器这样的设置为样式单的设置。

- h1：表示设置的是一级标题。以下为一级标题设置的情况；
- **color: green：设置色彩；**
- **font-style: italic：设置字体外观显示为斜体；**
- **font-size: 12pt：设置字体大小；**
- **h1.special：设置类；**
- **font-size：24pt；**
- **newheader：设置类；**
- **color: yello：设置色泽；**
- #caps：设置 id 属性；
- **font-variant: small-caps：设置字体显示；**
- </style>：结束样式单的定义；
- **<h1 class="newheader">Yellow, italic, and a point size of 12</h1>：类利用的标识；**
- **capitalized：设置段落的 id 属性。**

该页面在浏览器中显示的格式如图 16-1 所示。

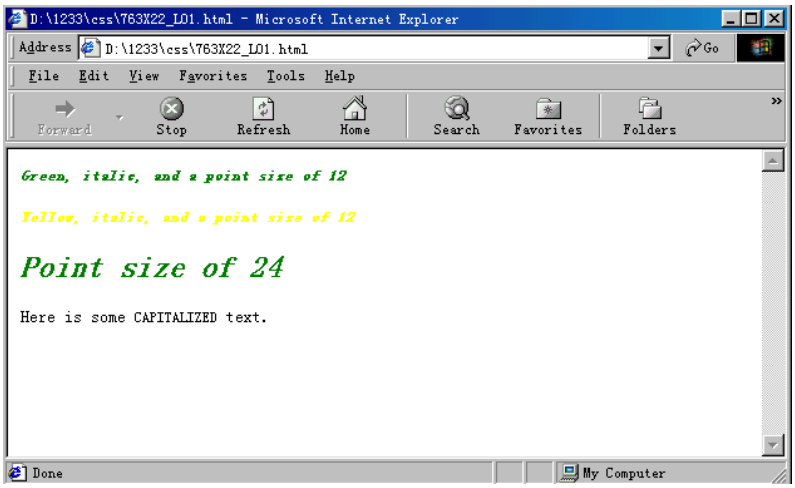


图16-1

通过以上的实例，可以看出样式单的定义可以精确地控制页面信息的显示。

16.44 样式单的定义

从上面的实例可以看到样式单的定义是从一个标志语句<style>开始的。在<style>中，

设置了如下语句格式：

<style type="text/css">

- **type：设置属性。这里设置样式单的属性，该属性的值为“text/css”。**

请看下边的实例：

```
    <html>
<head>
  <style type="text/css">
  <!--
    #heading{
      font-style: italic;
      font-size: 24pt;
    }
  -->
  </style>
</head>
<body>
  <p id="heading">
    Welcome to my page!
  </p>
  <p>
    Here is some text.
  </p>
</body>
</html>
```

在该实例中，设置了一个控制字体显示的样式单，设置字体的大小为 24 点，字体的外观显示为斜体。在浏览器中，页面的显示情况如图 16-2 所示。

其中，定义的样式单格式如下：

```
<style type="text/css">
  <!--
    #heading{
      font-style: italic;
      font-size: 24pt;
    }
  -->
  </style>
```

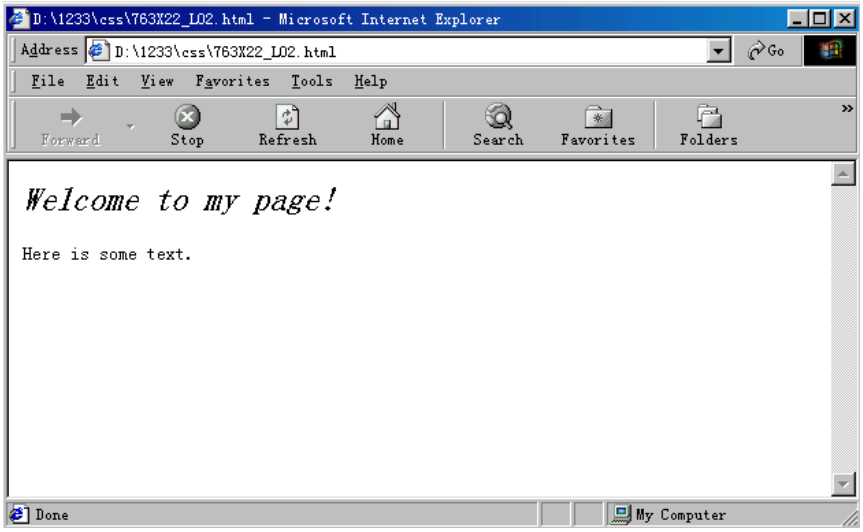


图16-2

在样式单的定义中，以<style>开头，以</style>结束样式单的定义。除此之外，还可以有如下定义的格式：

- **<link>：应用外部文档的样式单定义；**
- **：段落样式单的定义。**

16.45 样式单的使用

16.3.1 ids 属性与 id 属性

ids 属性用于设置 html 语言标志的属性。用户可以创建一个 id 设置 html 元素的显示。该语句的设置如下：

ids.id.property=value。

该属性的应用实例如下：

```
<html>
<head>
  <style type="text/css">
  <!--
    p{
      font-style: italic;
      font-size: 24pt;
    }
    #smaller{
      font-size: 12pt;
    }
```

```
    -->
    </style>
  </head>
  <body>
    <p>
      Here is some text that has global styles applied, but
      <span id="smaller">here is some smaller text</span> in
      the middle of this paragraph.
    </p>
  </body>
  </html>
```

在该页面中，设置的 ids 属性格式如下：

```
#smaller{
   font-size: 12pt;
}
```

在该格式中，设置字体的显示的大小为 12 点大小。

在 ids 属性的设置中，以“#smalle{”来设置格式的标头。然后在文档的使用中，为设置的 ids 属性指定一个 id 标识：

```
<span id="smaller">
```

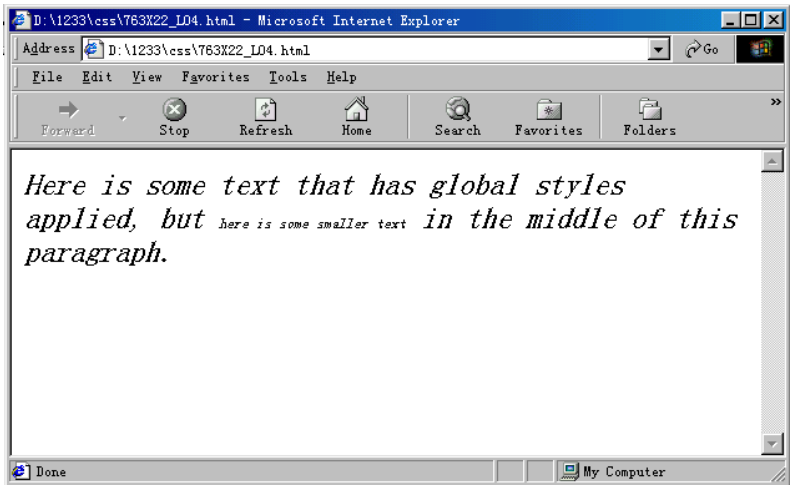


图16-3

该页面控制的显示效果如图 16-3 所示。

16.3.2 设置类 class 属性

class 控制文档元素类别显示格式。在应用中的格式与 ids 属性相同。在设置的过程中是以“.”开头来设置的。请看下边的实例。

```
<html>
<head>
  <style type="text/css">
  <!--
    h1{
      color: green;
    }
    #ital{
      font-style: italic;
    }
    .newheader{
      color: yellow;
    }
    #myid{
      font-size: 12pt;
```

```
    }
  -->
  </style>
</head>
<body>
  <h1>Green</h1>
  <h1 class="newheader" id="myid">Yellow and point size 12</h1>
  <h1 class="newheader">Yellow and <span id="ital">italic</span></h1>
  <h2>This will be default text<h2>
</body>
</html>
```

在该格式的定义中，设置了一个名为“newheader”的类属性，控制文本字体的显示色彩：

```
.newheader{
      color: yellow;
    }
```

在类的使用中，一般在文档元素中以关键字 class 指明类的应用，利用 class 关键字来规定文档应用到设置中的格式：

<h1 class="newheader">。

该页面的显示情况如图 16-4 所示。

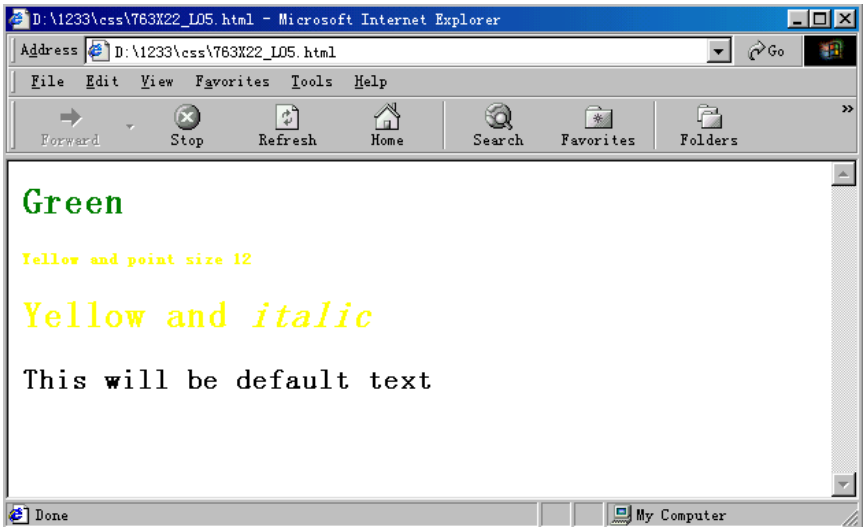


图16-4

16.3.3 tag 属　性

该属性用来设置文档 html 标记的样式单属性。一般用来指定一级标题的显示、文档主体的字体格式、段落的显示外观等等。

该格式在应用时，只要定义了 html 标记的属性，在文档中就自动使用设置的格式，无需用户再来设置。请看下边的实例：

```
    <html>
<head>
  <style type="text/css">
  <!--
```

```
        p{
            font-style: italic;
            font-size: 24pt;
        }
        #smaller{
            font-size: 12pt;
        }
    -->
    </style>
</head>
<body>
    <p>
        Here is some text that has global styles applied, but
        <span id="smaller">here is some smaller text</span> in
        the middle of this paragraph.
    </p>
</body>
</html>
```

该格式定义对段落 p 的显示外观的控制，并设置了段落字体显示为斜体，大小为 24 点。在文档显示时将自动执行设置的格式。在浏览器中显示的情况如图 16-5 所示。

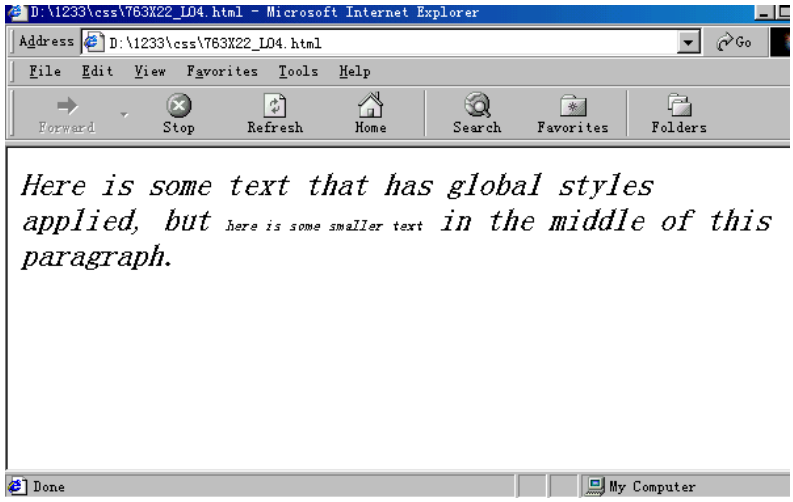


图16-5

第17章 实用小程序

主要内容

- 动态页面的概念
- 动态页面的实例

本章导读

本章主要通过具体的实例，介绍了如何在页面中设置可以利用的插入件和 Activex，以及利用 JavaScript 与插件和 ActiveX 通信。

17.46 导 言

到目前为止，我们已经深入地学习了 JavaScript 动态页面设计的知识。通过以上示例的学习，相信用户已经初步地掌握了 JavaScript 高级编程的知识。这一章，将介绍一些实用的例子，巩固前面学过的内容。

这一章我们将介绍如下的例子：

- 状态栏滚动信息;
- 计算用户来访次数;
- 散布在页面中的星星;
- 永在顶端的图片。

在本章的学习中，将列出示例的特性和源代码，并进行简单的分析。更具体的设置要由用户自己去分析。

17.47 状态栏滚动信息

在该示例中，设置了一个状态栏信息冒泡的效果。在静态的页面中，设置一个动态的状态栏信息，就会使页面更加吸引人。

这里利用 JavaScript 语言在状态栏模拟打字的效果。其脚本如下：

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>请看状态栏的变化</title>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
<script Language="JavaScript">
 var msg   = "欢迎你，我的朋友，让我们在 JavaScript 的世界中自由翱翔吧 ";
var interval = 100
var spacelen = 120;
var space10=" ";
var seq=0;
function scrollstr()
{
len = msg.length;
window.status = msg.substring(0, seq+1);
seq++;
if ( seq >= len )
seq = 0;
window.status = '';
window.setTimeout("scrollstr();", interval );
```

```
}
else
window.setTimeout("scrollstr();", interval );
}
scrollstr();
</script>
</head>

<body>

<p align="center"><big><strong><font color="#0000FF" face="楷体_GB2312"><big><big>请看状态栏的变化</big></big></font></strong></big></p>
</body>
</html>
```

其中：

- **var msg= "欢迎你，我的朋友，让我们在 JavaScript 的世界中自由翱翔吧"**：设置要在状态栏显示的用户信息;
- **函数 scrollstr()**：调用函数本身，每隔一秒，字符偏移一个位置。这样就构成了一个动态的打字效果;
- **window.status = msg.substring(0, seq+1)**：设置状态栏的信息;
- **window.setTimeout("scrollstr();", interval)**：调用 window 对象的 settimeout() 方法，调用设置的函数;
- **Interva**：设置 settimeout 超时的时间。

该脚本在 IE 中的表现如图 17-1 所示。

图17-80

17.48 计算用户来访次数

在该示例中，设置了一个脚本，用来处理用户来访次数，并最终将用户的数据显示在页面中。在 IE 中，页面的表现情况如图 17-2 所示。

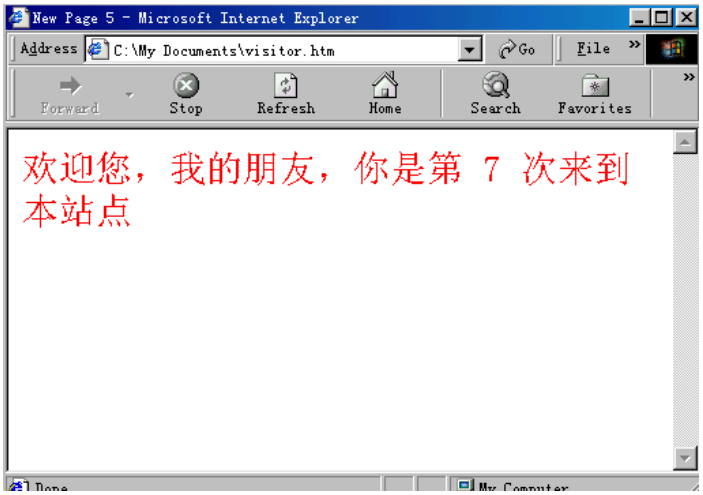


图17-81

实现如上页面的脚本源代码如下：

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>New Page 5</title>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
<script language="JavaScript">
<!--
var caution = false
function setCookie(name, value, expires, path, domain, secure) {
        var curCookie = name + "=" + escape(value) +
                ((expires) ? "; expires=" + expires.toGMTString() : "") +
                ((path) ? "; path=" + path : "") +
                ((domain) ? "; domain=" + domain : "") +
                ((secure) ? "; secure" : "")
        if (!caution || (name + "=" + escape(value)).length <= 4000)
                document.cookie = curCookie
        else
                if (confirm("Cookie exceeds 4KB and will be cut!"))
                        document.cookie = curCookie
}
function getCookie(name) {
        var prefix = name + "="
```

```
		var cookieStartIndex = document.cookie.indexOf(prefix)
		if (cookieStartIndex == -1)
			return null
		var cookieEndIndex = document.cookie.indexOf(";", cookieStartIndex + prefix.length)
		if (cookieEndIndex == -1)
			cookieEndIndex = document.cookie.length
		return unescape(document.cookie.substring(cookieStartIndex + prefix.length,
cookieEndIndex))
	}
	function deleteCookie(name, path, domain) {
		if (getCookie(name)) {
			document.cookie = name + "=" +
			((path) ? "; path=" + path : "") +
			((domain) ? "; domain=" + domain : "") +
			"; expires=Thu, 01-Jan-70 00:00:01 GMT"
		}
	}
	function fixDate(date) {
		var base = new Date(0)
		var skew = base.getTime()
		if (skew > 0)
			date.setTime(date.getTime() - skew)
	}
	var now = new Date()
	fixDate(now)
	now.setTime(now.getTime() + 730 * 24 * 60 * 60 * 1000)
	var visits = getCookie("counter")
	if (!visits)
		visits = 1
	else
		visits = parseInt(visits) + 1
	setCookie("counter", visits, now)
	document.write("<font size=6 color=red>欢迎您，我的朋友，你是第 " + visits + " 次来到本站点")
	// -->
	</script>
	</head>

	<body>

	<p align="right">  </p>
	</body>
	</html>
```

其中：

- **var caution = false**：设置一个逻辑变量;
- **function setCookie(name, value, expires, path, domain, secure)**：设置函数 setcookie()，该函数的设置与前面 cookie 文件设置一样，基础内容请看相关的章

节;

- **function getCookie(name)**: 设置函数 getcookie,检索用户设置的 name 值;
- **var cookieStartIndex = document.cookie.indexOf(prefix)**: 在检索函数中，调用字符串对象的 indexof()函数，看看设置的对象是否存在;
- **return null**: 如果用户设置的对象不存在，就返回一个 null 值;
- **function deleteCookie(name, path, domain)**: 设置 deletecookie（）函数，该函数处理文档 cookie 的过期时间;
- **function fixDate(date)**: 利用该函数来设置新文档 cookie 过期的时间 ，并在新文档中写入用户访问该页面的次数。

17.49 散布页面的星星

该示例说明动态页面中层的设置，用户在使用的过程中，可以看到一个跟随鼠标行进的星星。该示例侧重于利用 JavaScript 设置 jcss，即利用 JavaScrpt 语言设置页面的级联样式单，并控制页面格式的显示。

该页面在 IE 中的表现如图 17-3 所示。

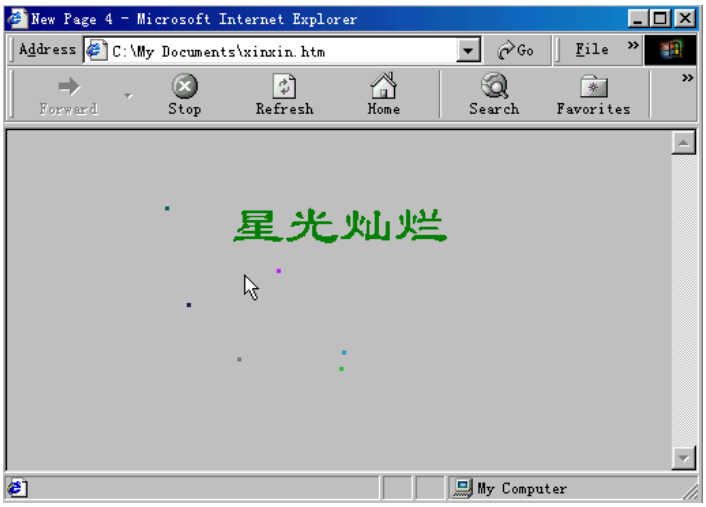


图17-82

实现如上页面的源代码如下：

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>New Page 4</title>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
<script>
```

```
if (!document.layers&&!document.all)
event="test"
function showtip2(current,e,text){

if (document.all&&document.readyState=="complete"){
document.all.tooltip2.innerHTML='<marquee style="border:1px solid black">'+text+'</marquee>'
document.all.tooltip2.style.pixelLeft=event.clientX+document.body.scrollLeft+10
document.all.tooltip2.style.pixelTop=event.clientY+document.body.scrollTop+10
document.all.tooltip2.style.visibility="visible"
}

else if (document.layers){
document.tooltip2.document.nstip.document.write('<b>'+text+'</b>')
document.tooltip2.document.nstip.document.close()
document.tooltip2.document.nstip.left=0
currentscroll=setInterval("scrolltip()",100)
document.tooltip2.left=e.pageX+10
document.tooltip2.top=e.pageY+10
document.tooltip2.visibility="show"
}
}
function hidetip2(){
if (document.all)
document.all.tooltip2.style.visibility="hidden"
else if (document.layers){
clearInterval(currentscroll)
document.tooltip2.visibility="hidden"
}
}

function scrolltip(){
if (document.tooltip2.document.nstip.left>=-document.tooltip2.document.nstip.document.width)
document.tooltip2.document.nstip.left-=5
else
document.tooltip2.document.nstip.left=150
}
</script>
<script LANGUAGE="JavaScript">
function YY_Layerfx(yyleft,yytop,yyfnx,yyfny,yydiv,yybilder,yyloop,yyto,yycnt,yystep) { //v1.2
//copyright (c)1999 Yaromat, Jaro von Flocken
  if ((document.layers)||(document.all)){
    with (Math) {yynextx= eval(yyfnx)}
    with (Math) {yynexty= eval(yyfny)}
    yycnt=(yyloop && yycnt>=yystep*yybilder)?0:yycnt+yystep;
    if (document.layers){
       eval(yydiv+".top="+(yynexty+yytop))
       eval(yydiv+".left="+(yynextx+yyleft))
    }
```

```
    if (document.all){
       eval("yydiv=yydiv.replace(/.layers/gi, '.all')");
       eval(yydiv+".style.pixelTop="+(yynexty+yytop));
       eval(yydiv+".style.pixelLeft="+(yynextx+yyleft));
    }
    argStr='YY_Layerfx('+yyleft+','+yytop+',"'+yyfnx+'","'+yyfny+'","'+yydiv+'",'
+yybilder+','+yyloop+','+yyto+','+yycnt+','+yystep+')';
    if (yycnt<=yystep*yybilder){eval(yydiv+".yyto=setTimeout(argStr,yyto)");}
  }

}
function YY_Mousetrace(evnt) { //v1.2 copyright (c)1999 Yaromat
  if (yyns4)
    {if (evnt.pageX) {yy_ml=evnt.pageX;    yy_mt=evnt.pageY;}    }
  else{
      yy_ml=(event.clientX + document.body.scrollLeft);
      yy_mt=(event.clientY + document.body.scrollTop);
  }
  if (yy_tracescript)eval(yy_tracescript)
}
</script>
<script language="JavaScript">
function PopWin()
{
    var PopWin = window.open("new.htm","PopWin","toolbar=no,directries=no,
scrollBars=yes,height=350,width=400");
}
</script>
</head>

<body text="#000000" link="#000080" vlink="#000080" bgcolor="#C0C0C0">
<div id="tooltip2"
style="position:absolute;visibility:hidden;clip:rect(0 150 50 0);width:150px;
background-color:lightyellow"><layer name="nstip" width="1000px" bgColor="lightyellow">
</layer>
</div>

<p align="center"><br>
<br>
</p>
<div id="yyd0"
style="position:absolute; left:10px; top:50px; width:3px; height:3px; z-index:1;
  background-color: #19636c; layer-background-color: #19636c; border: 1px none #000000;
clip: rect(0 3 3 0)"></div><div
id="yyd1"
style="position:absolute; left:20px; top:50px; width:3px; height:3px; z-index:1;
  background-color: #708574; layer-background-color: #708574; border: 1px none #000000;
  clip: rect(0 3 3 0)"></div><div
```

```
id="yyd2"
style="position:absolute; left:30px; top:50px; width:3px; height:3px; z-index:1;
background-color: #379bbf; layer-background-color: #379bbf; border: 1px none #000000;
clip: rect(0 3 3 0)"></div><div
id="yyd3"
style="position:absolute; left:40px; top:50px; width:3px; height:3px; z-index:1;
  background-color: #25184c; layer-background-color: #25184c; border:
1px none #000000; clip: rect(0 3 3 0)"></div><div
id="yyd4"
style="position:absolute; left:50px; top:50px; width:3px; height:3px; z-index:1;
background-color: #31bd3c; layer-background-color: #31bd3c; border:
1px none #000000; clip: rect(0 3 3 0)"></div><div
id="yyd5"
style="position:absolute; left:60px; top:50px; width:3px; height:3px; z-index:1;
background-color: #c11efd; layer-background-color: #c11efd; border:
1px none #000000; clip: rect(0 3 3 0)"></div><script>
var yyns4=window.Event?true:false; var yy_mt = 0; var yy_ml = 0;
document.onmousemove = YY_Mousetrace;
yy_tracescript = '';

if (yyns4){ document.captureEvents(Event.MOUSEMOVE);
   YY_Mousetrace('',',document.YY_Mousetrace1')}

YY_Layerfx(0,0,'yy_ml+cos((15*sin(yycnt/39.83007847812662))+0)*150*(sin(10+yycnt/20)+0.2)
*cos(yycnt/20)','yy_mt+sin((15*sin(yycnt/34.224861639800686))+0)*150*
(sin(10+yycnt/20)+0.2)*cos(yycnt/20)','document.layers[\'yyd0\']',2000,true,80,0,1);

YY_Layerfx(0,0,'yy_ml+cos((15*sin(yycnt/27.66510707209673))+30)*150*
(sin(10+yycnt/20)+0.2)*cos(yycnt/20)','yy_mt+sin((15*sin(yycnt/9.240632767417667))+30)
*150*(sin(10+yycnt/20)+0.2)*cos(yycnt/20)','document.layers[\'yyd1\']',2000,true,80,0,1);

YY_Layerfx(0,0,'yy_ml+cos((15*sin(yycnt/16.45318944579641))+60)*150*
(sin(10+yycnt/20)+0.2)*cos(yycnt/20)','yy_mt+sin((15*sin(yycnt/16.0564452288292))
+60)*150*(sin(10+yycnt/20)+0.2)*cos(yycnt/20)','document.layers[\'yyd2\']',2000,true,80,0,1);

YY_Layerfx(0,0,'yy_ml+cos((15*sin(yycnt/6.95348954836835))+90)*150*
(sin(10+yycnt/20)+0.2)*cos(yycnt/20)','yy_mt+sin((15*sin(yycnt/44.13697049887155))+90)*
150*(sin(10+yycnt/20)+0.2)*cos(yycnt/20)','document.layers[\'yyd3\']',2000,true,80,0,1);

YY_Layerfx(0,0,'yy_ml+cos((15*sin(yycnt/33.90077294583733))+120)*150*
(sin(10+yycnt/20)+0.2)*cos(yycnt/20)','yy_mt+sin((15*sin(yycnt/2.2378828869411587))+120)*
150*(sin(10+yycnt/20)+0.2)*cos(yycnt/20)','document.layers[\'yyd4\']',2000,true,80,0,1);

YY_Layerfx(0,0,'yy_ml+cos((15*sin(yycnt/37.858312521039835))+150)*150*
(sin(10+yycnt/20)+0.2)*cos(yycnt/20)','yy_mt+sin((15*sin(yycnt/18.083839795990098))+150)*
150*(sin(10+yycnt/20)+0.2)*cos(yycnt/20)','document.layers[\'yyd5\']',2000,true,80,0,1);
</script>
```

```
<p align="center"><big><font color="#008000" face="隶书"><big><big><big>星光灿烂
</big></big></big></font></big></p>
</body>
</html>
```

从这段源代码可以看出，这段程序大部分都是格式的控制。其中：

- **showtip2()函数**：该函数显示用户设置的滚动图标。事实上，该函数显示控制用户设置的 layer 的位置;
- **hidetip2()函数**：该脚本隐藏系统设置的层;
- **scrolltip()函数**：该函数设置层的相对位置;
- **YY_Layerfx()函数**：该函数设置层随鼠标的相对位置;
- **YY_Mousetrace()**：设置鼠标事件，捕获用户的鼠标运动过程;
- **PopWin()**：该函数在页面中开启一个新窗口，设置鼠标和星星的运动;
- **<div id="tooltip2"style="position:absolute;visibility:hidden;clip:rect(0 150 50 0);width:150px;background-color:lightyellow"><layer name="nstip" width="100- 0px" bgColor="lightyellow"**：该语句设置层的位置格式。

17.50 永在顶端的图片

该页面设置一个永在顶端的图片。在该页面中，无论用户向下移动多少文档，该图片永远在文档的顶端。在实际的利用中，这里的设置是相当有用的，对于较长的文档，我们可以设置这样的图片，在图片中设置一个锚点，跟随鼠标，当用户点击则回到页面的顶端。在该页面的设置中，我们只是实现了如上的功能，用户在设置中，可以不断地加以完善。

该页面在 IE 中的表现情况如图 17-4 所示。

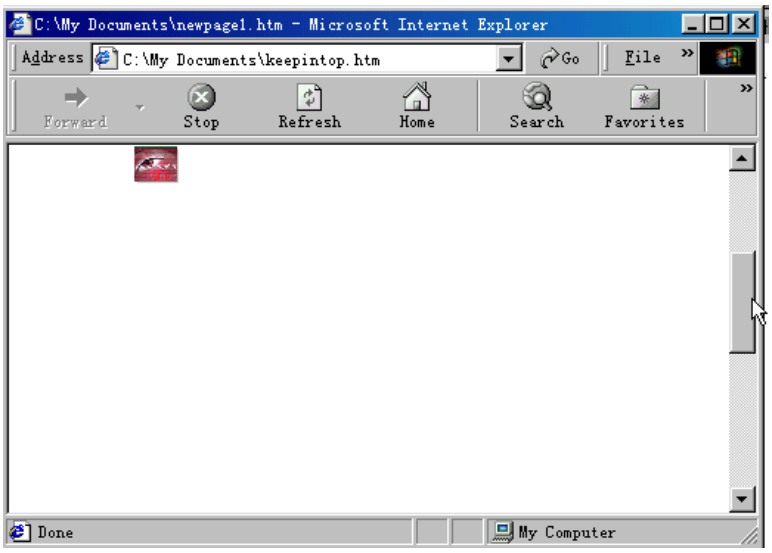


图17-83

该页面的源代码如下：

```
<html>

<head>
<title>C:\My Documents\newpage1.htm</title>
</head>

<body>
<div id="KBStatic"
style="position:absolute; left:600px; top:0px; width:200px; height:100px; z-index:25">

<p align="center"><img src="photo6.gif" width="30" height="26"
alt="photo6.gif (596 bytes)"></p>
</div><script language="JavaScript">
function keepintop(theName,theWantTop,theWantLeft) {
theRealTop=parseInt(document.body.scrollTop)
theTrueTop=theWantTop+theRealTop
document.all[theName].style.top=theTrueTop
theRealLeft=parseInt(document.body.scrollLeft)
theTrueLeft=theWantLeft+theRealLeft
document.all[theName].style.left=theTrueLeft
}
function keepinIE(theName,theWantX,theWantY) {
theRealLay=document.layers[theName]
theBadX=self.pageYOffset
theBadY=self.pageXOffset
theRealX=theBadX+theWantX
theRealY=theBadY+theWantY
theRealLay.moveTo(theRealY,theRealX)
}
IE4=(document.all)?1:0
NN4=(document.layers)?1:0
if (IE4)
setInterval('keepintop("KBStatic",0,0)',1)
if (NN4)
setInterval('keepinIE("KBStatic",0,0)',1)
</script>


<p>  </p>

<p>  </p>

<p>  </p>

<p>  </p>

<p>  </p>
```

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

其中：

- **<div id="KBStatic"style="position:absolute; left:600px; top:0px; width:200px; height:100px; z-index:25">**: 该语句设置层的格式。用心的读者可以去查看相关的书籍。

从以上的示例中，我们复习巩固以前学的一些知识，相对来说，这些示例在巩固以前学的知识的过程中，有了一些深入的东西，这里虽然只是列出了他们的源代码而没有做深入的讲解，就是希望用户在利用的过程中，自己试着分析，这样学习才会进步。

无论怎样，我们衷心的希望通过本书的学习，使你的脚本编写能力有一个长足的进步，完善你设计的页面。

熟练地掌握 JavaScript 并不是一朝一夕的事情，你还得在实际的工作中不断地完善不

断地提高。但是笔者相信，通过本书基础部分和示例的学习，你应该具有良好的 JavaScript 基础了，并能够胜任一般脚本编程的开发。

第18章 JavaScript 语言的扩展

主要内容

- ActiveX 通　信
- 调用插入件

本章导读

利用 HTML 语言来开发页面的能力是有限的。在所谓的静态页面领域里，HTML 语言的扩展已经到了尽头。为了使页面设计更加专业和简单化，为了使设计的页面更加丰富，在越来越多的扩展标准中，加入了 HTML 语言与 Java 小程序、ActiveX、插入件通信的能力。在这些语言的扩展中，JavaScript 脚本也设置了与这些扩展通信的能力，以便利用脚本语言来控制这些物件的显示、关闭、调用等等。

在本章中，我们利用一些实例来学习 JavaScript 与 Java 小程序，ActiveX 和插入件通信的能力。考虑到一般用户的设置要求，这里不讲述与 Java 小程序通信。下面来看看 JavaScript 与 ActiveX 的脚本设置情况。

18.51 ActiveX 通 信

18.51.1 ActiveX 简 述

ActiveX 是为 windows 3.1 开发的对象连接与嵌入和组件对象模型技术的扩展。ActiveX 是开发和使用 Microsoft 所生成的软件组件方法。Microsoft 在 1996 年引入了 ActiveX 的意图就是让 com 组件在 web 领域上利用 Internet 控件。

18.51.2 利用 ActiveX 实例

程序源代码如下：

```
<HTML>
<head>
<title>JavaScript Unleashed</title>
<script type="text/JavaScript">
<!--
function loadBox()
{
alert("The Great Question");
}
//-->
</script>
<script type="text/JavaScript" for="CommandButton1" event="Click()">
<!--
var msg = "She loves me";
var altMsg = "She loves me not";
if(CommandButton1.Caption == msg)
{
CommandButton1.Caption = altMsg;
window.defaultStatus = altMsg;
}
else
{
if(CommandButton1.Caption == altMsg)
{
CommandButton1.Caption = msg;
window.defaultStatus = msg;
```

```
}
}
//-->
</script>
</head>
<body onload="loadBox()">
<object id="CommandButton1" width="98" height="32" classid="CLSID:D7053240-CE69-11CD- A777-
00DD01143C57">
<param name="VariousPropertyBits" value="268435483">
<param name="Caption" value="She loves me">
<param name="Size" value="2096;678">
<param name="FontCharSet" value="0">
<param name="FontPitchAndFamily" value="2">
<param name="ParagraphAlign" value="3">
<param name="FontWeight"    value="0">
</object>
</body>
</HTML>
```

18.51.3 程 序 简 介

1. ActiveX 语　法

在这段程序中，包含了脚本语句以及 ActiveX 控件语句。其中以<object></object>为标志的 z 标记为 ActiveX 格式语句。如下：

```
<object id="CommandButton1" width="98" height="32" classid="CLSID:D7053240-CE69-11CD-
A777-00DD01143C57">
<param name="VariousPropertyBits" value="268435483">
<param name="Caption" value="She loves me">
<param name="Size" value="2096;678">
<param name="FontCharSet" value="0">
<param name="FontPitchAndFamily" value="2">
<param name="ParagraphAlign" value="3">
<param name="FontWeight"    value="0">
```

- **</object>**：对于 ActiveX 语句是以 id 来标记的;
- id="CommandButton1"：标识的 ID 属性提供了 ActiveX 控件的标识符号，可以用它在脚本中调用设置的控件;
- classid="CLSID:D7053240-CE69-11CD-A777-00DD01143C57"：该属性用来标识惟一的 ActiveX 对象的类;
- <param name="VariousPropertyBits" value="268435483">：利用<param>来标识 ActiveX 组件的参数名和参数值。

2. 脚 本 语 句

对于 ActiveX 控件的设置就如上面的一样。下面来看看页面设置的脚本。在这段程序的脚本中，引入了不同于以往的设置——引入了关键字 type：

<script type="text/JavaScript">

这种标记为 JavaScript 1.5 所支持的结构，利用这种标记，可以使脚本语言的结构设置与 HTML 语言的设置一致，同时，这种变化也得到浏览器的支持。

- loabbox()函数

该函数在文本中写入一句话“the great question”。在设置的过程中，调用 window 对象的 alert()方法，设置提示信息。脚本语句如下：

```
function loadBox()
{
alert("The Great Question");
}
```

- 利用脚本访问 ActiveX 控件

在这段脚本语句中，设置了一个循环，当用户按下设置的按钮时，就周而复始地显示用户设置的信息。在这段脚本中，设置了利用脚本与控件通信的方法。

脚本如下：

```
<script type="text/JavaScript" for="CommandButton1" event="Click()">
<!--
var msg = "She loves me";
var altMsg = "She loves me not";
if(CommandButton1.Caption == msg)
{
CommandButton1.Caption = altMsg;
window.defaultStatus = altMsg;
}
else
{
if(CommandButton1.Caption == altMsg)
{
CommandButton1.Caption = msg;
window.defaultStatus = msg;
}
}
//-->
</script>
```

其中：

- <script type="text/JavaScript" for="CommandButton1" event="Click()">: 设置脚本语句的种类为 JavaScript，控制的对象为设置的 ActiveX 控件 commandbutton1，事件的处理为点击 click();
- var msg = "She loves me": 设置变量;
- if(CommandButton1.Caption == msg): 设置判断条件。该语句的意义为如果按

钮的文字为设置的第一个字符串变量，就执行下面的语句;

- **CommandButton1.Caption = altMsg:** 重新设置按钮的值;
- **window.defaultStatus = altMsg:** 设置窗口状态条的信息。

这样通过简单的调用，就设置了利用 JavaScript 与 ActiveX 控件通信的方法。该程序在浏览器中的显示情况如图 18-1 所示。

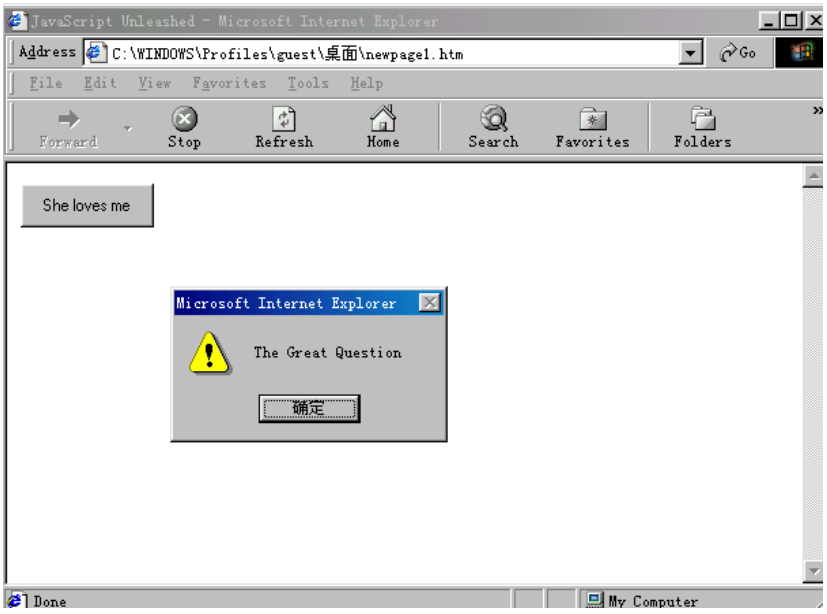


图18-84

当点击之后，就会出现如图 18-2 所示的页面。

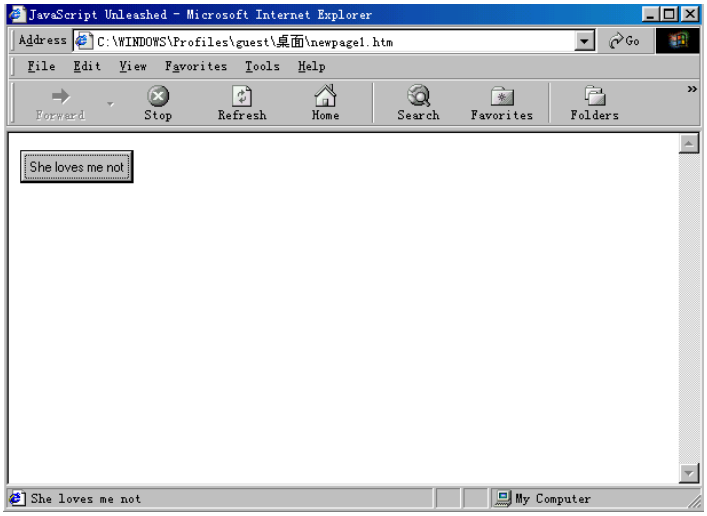


图18-85

18.52 调用插入件

插入件的利用也是 HTML 语言扩展的技术之一，利用插入件，可以轻易地实现多媒体页面的设计。常用的插入件为 liveaudio 和 quicktime、flash。

在脚本设置的过程中，可以设计脚本语言来控制对插入件的调用。事实上，对插入件的控制属于插入件的设置者，只能利用 JavaScript 脚本语言来设置与插入件进行交互的操作。

在如下的程序中，设置了利用 JavaScript 脚本来控制音乐文件的播放与停止，读者在使用程序的过程中，将音乐文件换成本地的音乐文件就可以了。程序的代码如下：

```
<html>
<head>
  <title>JavaScript Unleashed</title>
  <script language="javascript">
  <!--
    function playSound(sfile) {

      // load a sound and play it
      window.location.href=sfile;
    }
  //-->
  </script>
</head>
<body onLoad="playSound('test.wav')" onUnLoad="playSound('test.wav')">
  <h2>
    Sounds on JS Events
  </h2>
  <hr>
  The following are example of JS event handlers used to play sounds.
  <hr>
  <a href="#" onClick="playSound('test.wav')">
    Click here for sound
  </a>
  <p>
    <form name="form1">
      <input type="button" value="Play" onClick="playSound('test.wav')">
    </form>
  </p>
</body>
</html>
```

其中：

设置插入件的语句格式如下：

- **<embed mastersound name="sound1" src="test.wav" volume="100" hidden= "true"autostart="false">**：该语句有一个标识<embed>，在该标识之间必须包含一个 src 属性或 type 属性。这里 src 指明声音文件的来源。name 属性设置一个插入件的标识，有助于在脚本语句中调用。

随后在这段程序中，设置了简单的脚本语句来调用设置的插入件：

```
<form name="form1">
<input type="button" value="Play" onclick="document.sound1.play(true)">
<input type="button" value="Pause" onclick="document.sound1.pause()">
<input type="button" value="Stop" onclick="document.sound1.stop()">
</form>
```

设置的插入件的调用通过在表单按钮的点击来完成的。如果用户要播放设置的插入件音乐，就设置插入件的播放属性为 true。否则相应的设置为 pause 或 stop。这样通过简单的调用就设置了对插入件的交互。

在浏览器中，该脚本的显示如图 18-3 所示。

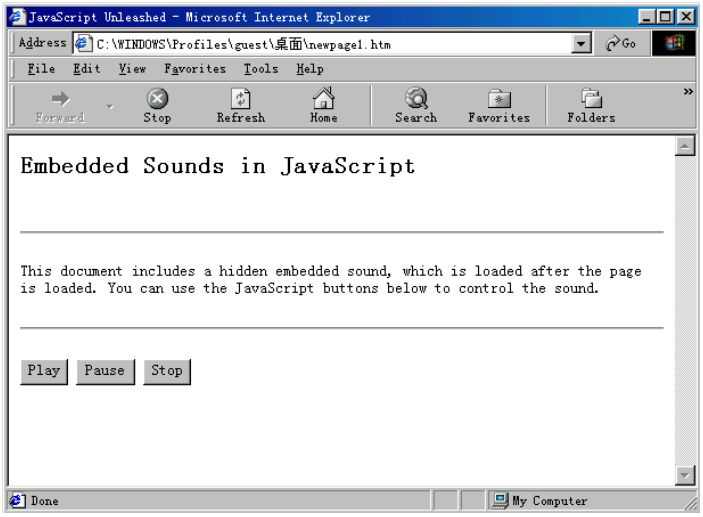


图18-86

当点击相应的按钮时，就会听到音乐的效果了。

第19章 网上购物系统

主要内容

- 示 例 特 性
- 源 代 码

本章导读

在本示例中，利用 JavaScript 和 HTML 语言，开发一个网上购物系统。在该购物系统中，没有复杂的 CGI 程序，完全由 JavaScript 和 HTML 语言构成。在该示例中，设置了一些列脚本，实现了表单与用户交互，实现了利用表单与用户通信，最后通过电子邮件，将用户的需求传递给相关人员进行相应的处理。

事实上，示例的开发仅是一个联机购物系统，包括了点击选择产品、自动计算总量、收集客户信息、显示用户定货信息，最后通过电子邮件的形式提交表单，这样避免了复杂的 CGI 编程。

对于中小企业来说，这样的联机购物系统是相当有效的，编程简单、维护方便，对硬件要求低，只要拥有一个电子邮件信箱地址，并连通网络，定时收发电子邮件并进行处理就可以实现网络营销，以最小的投入获得最大的利润。

19.53 示例特性

该示例为中小企业实现电子商务服务，在设置的过程中设置了三个帧组，将窗口分为有机的三个部分，便于系统管理人员维护与管理。通过该示例的学习，用户可以实现复杂的脚本编写技术、参数和函数的调用。

该示例有如下特性：

- 帧组的设置;
- 表单的设计;
- 窗口对象的方法;
- mailto 协议的设置;
- 字段的验证;
- 数据初始化;
- 函数的设置;
- 参数的传递;
- 函数的调用;
- 复杂的嵌套;
- 数据的自动格式化;
- 表单事件处理器的设置方法、技巧;
- 利用 HTML 和 JavaScript 实现联机购物系统的设置;
- 组合 HTML、JavaScript、浏览器成常用的 web 页面应用程序。

19.54 源代码

在实现如上特性的 web 页面中，系统设置了三个帧，将窗口分割为相互独立的区域。第一个帧设置页面的题头，在第二帧中设置产品目录，在第三个帧中设置客户信息并显示客户的定货信息。

这里先将设置的源代码列出来，用户可以先试着分析一下，在详解程序的章节中，我们将重点讲解。

父帧将窗口分为三个帧，并设置相应页面，其源代码如下：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>

<head>
<title>welcome</title>
<meta content="text/HTML; charset=gb2312" http-equiv="Content-Type">
<meta content="Microsoft FrontPage 3.0" name="GENERATOR">
</head>

<frameset rows="64,*">
  <frame name="banner" noResize scrolling="no" src="New%20Page%203.htm"
  style="BACKGROUND-COLOR: rgb(255,0,255)" target="contents">
  <frameset cols="251,*">
    <frame name="contents" src="New%20Page%204.htm" target="main" scrolling="auto">
    <frame name="main" src="New%20Page%205.htm" scrolling="auto">
  </frameset>
  <noframes>
  <body>
  </body>
  </noframes>
</frameset>
</HTML>
```

在 banner 帧中，设置文档的标题，其源代码如下：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>

<head>
<title>New Page 3</title>
<meta content="text/HTML; charset=gb2312" http-equiv="Content-Type">
<meta content="Microsoft FrontPage 3.0" name="GENERATOR">
<meta content="none" name="Microsoft Border">
<base target="contents">
</head>

<body>

<p align="center"><font color="#0000ff" face="华文新魏"><big><big><big><big>网上超市
</big></big></big></big></font></p>
</body>
</HTML>
```

在 contents 帧中设置产品目录，并列出销售的产品，让用户点击选择产品。其源代码如下：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>

<head>
```

```
<title>New Page 4</title>
<meta content="text/HTML; charset=gb2312" http-equiv="Content-Type">
<meta content="Microsoft FrontPage 3.0" name="GENERATOR">
<script language="JavaScript">
<!--
var youjifei=0,huokuan=0,totalfei=0;

function isnumber(theinput)
{

for(var i=0;i<theinput.length;i++)
{
var isnum=theinput.substring(i,i+1);
if(isnum<"0"||isnum>"9")
{
alert("请输入数字！！！")
return false;
}
else
{
return true;
}
}
}
function choosemorden()
{
if(menu.morden.checked==true)
{
var input=prompt("请输入订购的数量");
if(input==""||input=="undefined"||input==null)
{
menu.morden.checked=false;
alert("请输入数据");
return false;
}
else
{
if(isnumber(input)==true)
{
if(confirm("单价：200 元\n 订购数量:"+input+"\n 货款合计为："+eval(200*eval(input))+"\n 邮寄费
为："+eval(eval(200*eval(input))*0.01)))
{menu.morden.value=input;
youjifei+=eval(200*eval(input))*0.01;
huokuan+=eval(200*eval(input));
parent.main.document.order.S1.value+="\n 全向激光 morden          数量："+input+"\n";
parent.main.document.order.fei.value=youjifei;
parent.main.document.order.trade.value=huokuan;
parent.main.document.order.total.value=eval(parent.main.document.order.trade.value)+eval(parent.main.
```

```
document.order.fei.value);
    return true;
    }
    else
    {
    menu.morden.checked=false;
    menu.morden.value="";
    }
    }
    }
    }
    }

    function chooseprint()
    {
    if(menu.print.checked==true)
    {
    var input=prompt("请输入订购的数量");
    if(input==""||input=="undefined"||input==null)
    {
    menu.print.checked=false;
    alert("请输入数据");
    return false;
    }
    else
    {
    if(isnumber(input)==true)
    {

    if(confirm("单价：5000 元\n 订购数量:"+input+"\n 货款合计为："+eval(5000*eval(input))+"\n 邮寄
费为："+eval(eval(5000*eval(input))*0.01)))
    {menu.print.value=input;
    youjifei+=eval(5000*eval(input))*0.01;
    huokuan+=eval(5000*eval(input));
    parent.main.document.order.S1.value+="\nLM-AS 打印机                数量："+input+"\n";
    parent.main.document.order.fei.value=youjifei;
    parent.main.document.order.trade.value=huokuan;
    parent.main.document.order.total.value=eval(parent.main.document.order.trade.value)+eval(parent.main.
document.order.fei.value);
    return true;
    }
    else
    {
    menu.print.checked=false;
    menu.print.value="";
    }
    }
    }
```

```
}

}
function choosephoto()
{
if(menu.photoshop.checked==true)
{
var inputprint=prompt("请输入订购的数量");
if(inputprint==""||inputprint=="undefined"||inputprint==null)
{
menu.photoshop.checked=false;
alert("请输入数据");
return false;
}
else
{
if(isnumber(input)==true)
{

if(confirm("单价：25 元\n 订购数量:"+inputprint+"\n 货款合计为："+eval(25*eval(inputprint))+"\n
邮寄费为："+eval(eval(25*eval(inputprint))*0.1)))
{menu.photoshop.value=inputprint;
youjifei+=eval(25*eval(inputprint))*0.1;
huokuan+=eval(25*eval(inputprint));
parent.main.document.order.S1.value+="\nphotoshop6.0          数量："+inputprint+"\n";
parent.main.document.order.fei.value=youjifei;
parent.main.document.order.trade.value=huokuan;
parent.main.document.order.total.value=eval(parent.main.document.order.trade.value)+eval(parent.main.
document.order.fei.value);
return true;
}
else
{
menu.photoshop.checked=false;
menu.photoshop.value="";
}
}
}
}
}
// --></script>

<base target="main">
</head>

<body>
<div align="left">
```

```
<table border="1" cellPadding="0" cellSpacing="0" height="18"
style="BORDER-BOTTOM: rgb(255,0,255) 1px groove; BORDER-LEFT: rgb(0,0,128) 1px groove;
BORDER-RIGHT: rgb(0,0,128) 1px groove; BORDER-TOP: rgb(255,0,255) 1px groove"
width="158">
<TBODY>
  <tr>
    <td align="middle" height="18" width="158"><p align="center"><font face="楷体
_GB2312"><big>产品目录</big></font></td>
  </tr>
</TBODY>
</table>
</div>

<p><br>
<img alt="mood3.gif (88 bytes)" height="15" src="mood3.gif" width="15">请点击选择的产品<br>
<img alt="mood3.gif (88 bytes)" height="15" src="mood3.gif" width="15">请清楚填写产品数目</p>

<hr SIZE="1">

<form name="menu">
  <p><input name="morden" onclick="choosemorden()" type="checkbox" value="ON"><img
  alt="morden.gif (289 bytes)" src="morden.gif" width="30" height="20">全向激光 morden<br>
  <input name="print" onclick="chooseprint()" type="checkbox" value="print"><img
  alt="print.gif (1149 bytes)" src="print.gif" width="40" height="35">LM-AS 打印机<br>
  <input name="photoshop" onclick="choosephoto()" type="checkbox" value="photoshop"><img
  alt="photo6.gif (596 bytes)" src="photo6.gif" width="30" height="26">photoshop6.0</p>
</form>
</body>
</HTML>
```

在 main 帧中，设置表单获得用户信息和用户订购信息。最后以 email 的形式发送到相关人员的邮箱中进行处理。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>

<head>
<title>New Page 5</title>
<meta content="text/HTML; charset=gb2312" http-equiv="Content-Type">
<meta content="Microsoft FrontPage 3.0" name="GENERATOR">
<meta content="none" name="Microsoft Border">
<script language="JavaScript">
<!--
function isrequired(thefield)
{
if( thefield.value=="")
{
alert("请输入数据！！！");
return false;
```

```
}
else
{
return true;
}
}
function isnumber(theinput)
{
if(isrequired(theinput)==true)
{
for(var i=0;i<theinput.value.length;i++)
{
var isnum=theinput.value.substring(i,i+1);
if(isnum<"0"||isnum>"9")
{
alert("请输入数字！！！")
return false;
}
else
{
return true;
}
}
}
}
function isemail(theinput)
{
if(isrequired(theinput)==true)
{
var isnum=theinput.value.indexOf("@");
if(isnum!=-1&&theinput.value.indexOf(".")!=-1)
{
alert("thank！！！");
return true;
}
else
{
alert("请输入正确的邮箱地址！！");
theinput.value="";
return false;
}


    }
}
function iszip(theinput)
{
if(theinput.value.length!=6)
```

```
{
alert("邮编错误");
theinput.value="";
return false;
}
else
{
isnumber(theinput);
}
}

function isdate(theinput)
{
if(isrequired(theinput)==true)
{
var thechar=theinput.value.substring(4,5);
if(thechar!="/")
{
alert("请输入正确的日期格式(如 2000/12/25）");
return false;
}
else
{
if(confirm("交货日期为："+theinput.value+"。"))
{
alert("谢谢你，我们将及时与你联系，准时交货！！")
}
else
{
theinput.value="";
isrequired(theinput);
}
}
}
}
function quxiao()
{
parent.contents.document.menu.morden.checked=false;
parent.contents.document.menu.print.checked=false;
parent.contents.document.menu.photoshop.checked=false;
alert("请重新选取物品！");
}

function queren()
{
if(order.date.value==""||order.adress.value==""||order.name.value==""||order.trade.value=="0")
{
order.action="";
```

```
alert("请填写好用户信息，选择产品，填上该产品的数量\n 确认后，再送出该数据！ ");

return false;
}
else
{
alert("谢谢你订购本公司的产品！ ");
}
}
// -->
</script>
</head>

<body>

<form action="mailto:tsg@mail.scit.com" method="post" name="order">
  <p>请客户填写如下信息：<br>
  <br>
  姓名：<input name="name" onblur="isrequired(order.name)" size="8">电话：<input
  name="phone" onblur="isnumber(order.phone)" size="10">电子信箱：<input name="email"
  onblur="isemail(order.email)" size="12"><br>
  地址：<input name="adress" onblur="isrequired(order.adress)" size="34">邮编：<input
  name="zip" onblur="iszip(order.zip)" size="7"><br>
  你的信用卡种类：<input CHECKED name="R1" type="radio" value="jianhang">建行信用卡<input
  name="R1" type="radio" value="nonghang ">农行信用卡<input name="R1" type="radio"
  value="gonghang">工行信用卡<br>
  请输入卡号：<input name="cardnum" onblur="isnumber(order.cardnum)" size="20">交货日期<input
  name="date" onblur="isdate(order.date)" size="12" value="2000/3/12"><br>
  <br>
  请查看你选择的产品信息：<br>
  <br>
  <textarea cols="30" name="S1" rows="4">以下是你的订购信息：</textarea><br>
  邮寄费：<input name="fei" size="10" value="0">货款合计<input name="trade" size="8"
  value="0"><br>
  总计：<input name="total" size="20" value="0.00"></p>
  <p><input name="B1" onclick="queren()" type="submit" value="确认订购"><input name="B2"
  onclick="quxiao()" type="reset" value="取消订购"></p>
</form>
</body>
</HTML>
```

19.55 功能概述

本例的调试工作是在IE中进行的。一项调查显示，国内的操作系统平台是基于windows的，而IE浏览器是捆绑销售的，而基于IE浏览器的优良性能，国内的大多用户使用的是

IE 浏览器。该示例能很好地在 IE 中运行，相信在其他浏览器中也能很好地运行。

在 IE 中，该程序的表现情况如图 19-1 所示。

可以看出，该窗口被滚动条分为三个帧，分别为题头帧、产品目录帧、客户信息帧。在设置的产品目录帧中，点击设置的销售产品前面的复选框，这时就会弹出一个对话框，提示用户输入产品数量，如图 19-2 所示。

如果用户没有输入任何数据而直接按下【OK】或【Cancel】按钮，系统就会弹出一个提示信息框，让用户输入数据，并取消刚才对产品的选择，如图 19-3 所示。

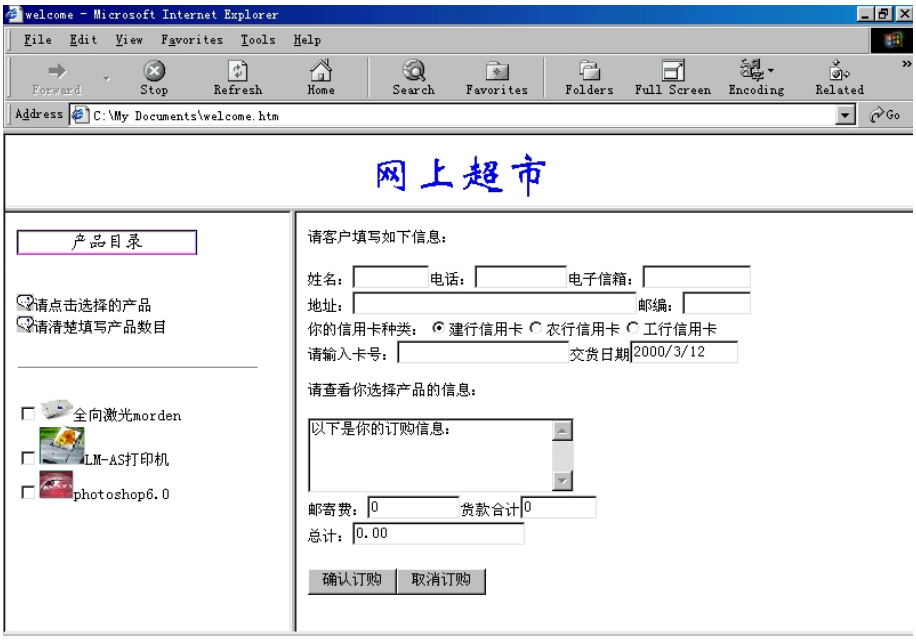


图19-1

图19-2

图19-3

如果用户输入的数据不是数字，通过系统的检测，就会提示用户输入数字,如图 19-4、图 19-5 所示。

图19-4

如果用户输入的数据正确，就会弹出一个对话框，显示用户的订购情况，让用户确认，如图 19-6 所示。

图19-5

图19-6

如果用户认为订购的数量不合适或其他方面的原因要取消该订购，按下【取消】按钮，这时就会取消对该产品的选择。

如果用户按下【确定】按钮，就会在客户信息区显示用户的订购情况。在设置的滚动文本框中显示用户的订购产品名称和数量，在邮寄费中显示邮寄费，在货款中显示货款合计，在总计中显示成交这笔交易要付款的总额，如图 19-7 所示。

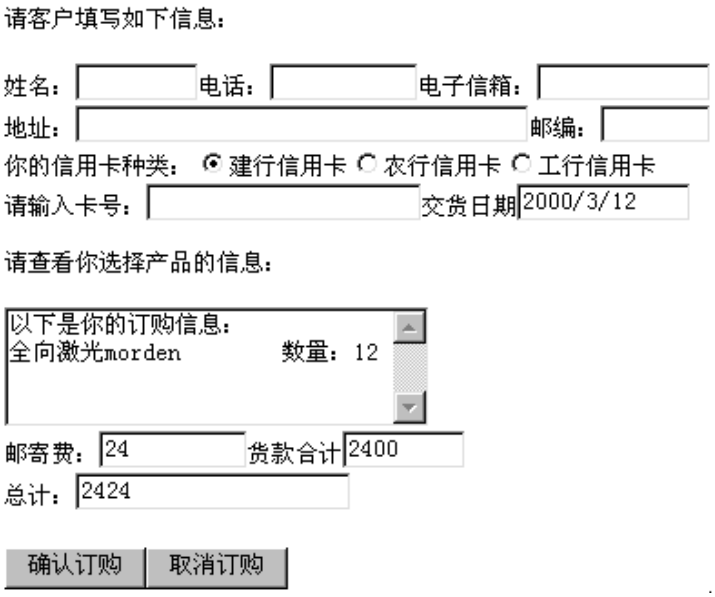


图19-7

在设置了产品之后，还要求填写客户信息，如果用户没有填写客户信息，系统认为这是一种破坏和恶作剧的行为，提示用户应该填写完，否则会取消对产品的选择，如图 19-8、图 19-9 所示。

图19-8

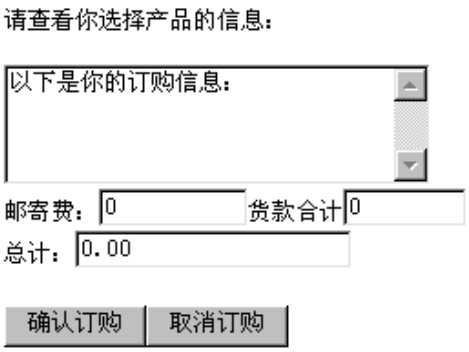


图19-9

在填写客户信息的时候，设置了一些过滤条件、必填字段，比如姓名，如果用户不输入姓名，则提示用户输入数据，如图 19-10 所示。

图19-10

如果用户在输入电话的过程中，输入的电话号码不正确，则提示用户输入正确的电话号码。这里判断的依据就是看用户输入的数据是否全为数字,如图 19-11。

图19-11

在设置用户输入电子邮箱的过程中，设置的判断条件就是看用户输入的数据中是否包含“@”和“.”。如果没有，就认为输入得不正确，如图 19-12 所示。

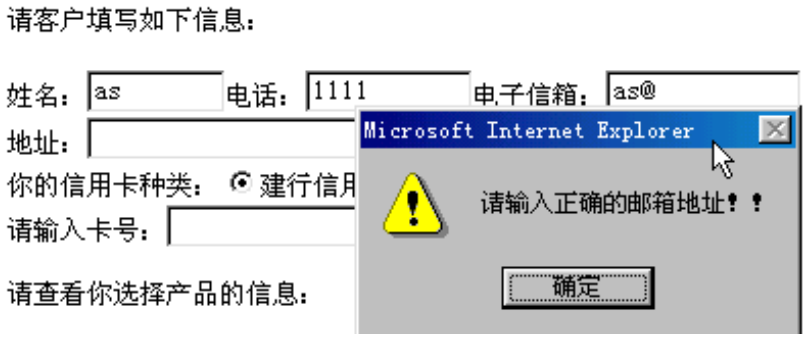


图19-12

在交货日期的设置中，设置了一个提示格式，用户只能按提示的格式输入，否则认为是非法输入，如图 19-13 所示。

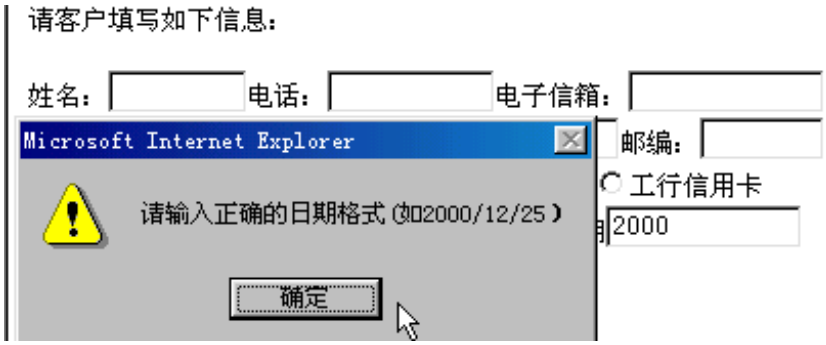


图19-13

如果用户输入正确，则弹出一个确认信息框，让用户确认，如果用户确认后，就会弹出如图 19-14、图 19-15 所示的提示框。

图19-14

图19-15

如果用户按下【取消】按钮，则会清除用户的输入，提示用户输入数据，如图 19-16 所示。

在填写完毕后，如果确认无误，按下【确认订购】按钮，这时系统就会启动系统的 email 系统，提交用户的订购信息，如图 19-17、图 19-18 所示。

图19-16

图19-17

如果用户点击【取消订购】按钮，就会清除用户的订购信息，提示用户输入数据,如图 19-19 所示。

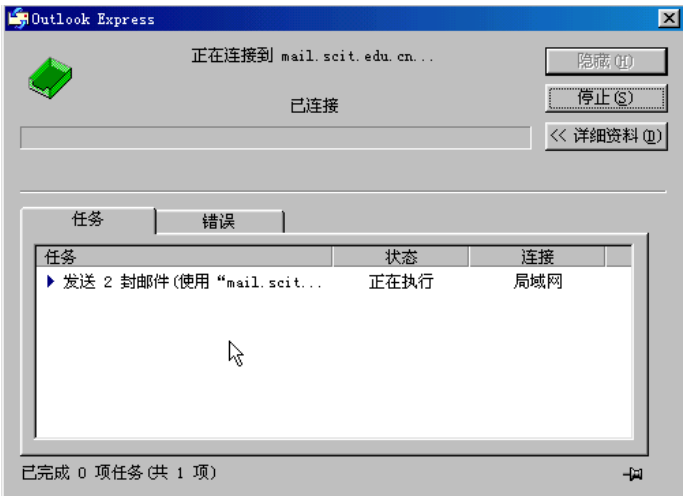


图19-18

图19-19

19.56 程序详解

在父帧中，设置了利用帧将窗口分割。在每个帧中，设置相应的属性值。其中：

- **name 属性**：设置了帧的名称属性，在设置了该属性后，在脚本调用中显得相当方便。
- **Src**：设置帧页面属性，在该帧中显示由 src 指定的页面。

同时，通过该帧组的设置，启发用户在设置相应的应用程序的过程中，可以将窗口分割，将相关的页面放置在不同的帧中。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>

<head>
<title>welcome</title>
<meta content="text/HTML; charset=gb2312" http-equiv="Content-Type">
<meta content="Microsoft FrontPage 3.0" name="GENERATOR">
</head>

<frameset rows="64,*">
  <frame name="banner" noResize scrolling="no" src="New%20Page%203.htm"
  style="BACKGROUND-COLOR: rgb(255,0,255)" target="contents">
  <frameset cols="251,*">
    <frame name="contents" src="New%20Page%204.htm" target="main" scrolling="auto">
    <frame name="main" src="New%20Page%205.htm" scrolling="auto">
  </frameset>
  <noframes>
  <body>
  <p>如果你看到这段文字，则说明你的系统不支持帧组，请升级你的浏览器。</p>
  </body>
  </noframes>
</frameset>
</HTML>
```

其中：

- **<frameset rows="64,*">**：设置帧组，将窗口分为上下两部分。上帧的高度为 64 个单位;
- <frame name="banner" noResize scrolling="no" src="New%20Page %203.htm" style="BACKGROUND-COLOR: rgb(255,0,255)" >：设置帧组名为 “banner”，不能改变大小，不显示滚动条，显示的页面为同级目录下的 new page3.htm 文档，该帧的背景为白色;
- **<frameset cols="251,*">**：将第二帧纵向分割，将出口分为两列。设置第二帧的大小为 251 个单位;
- <frame name="contents" src="New%20Page%204.htm" target="main" scrolling

="auto">：设置帧名为“contents”，显示的页面为“newpage4.htm”，目标帧为‘main’。可以自动滚动设置窗口的大小;

- **<noframes>**：设置处理浏览器不支持帧组的情况。这里提示用户要进行浏览器升级的设置。

19.4.1 Banner 帧

在 banner 帧中，设置显示页面的标题。用户在设置的时候，还可以将一些相关的页面信息放置在该页面中，比如导航条，链接信息，以及相应的日期信息等等。该例的设置简单，显示的效果如图 19-20 所示。

图19-20

该程序的源代码如下：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>

<head>
<title>New Page 3</title>
<meta content="text/HTML; charset=gb2312" http-equiv="Content-Type">
<meta content="Microsoft FrontPage 3.0" name="GENERATOR">
<meta content="none" name="Microsoft Border">
<base target="contents">
</head>

<body>

<p align="center"><font color="#0000ff" face="华文新魏"><big><big><big><big>网上超市
</big></big></big></big></font></p>
</body>
</HTML>
```

其中：

- <p align="center">：设置段落居中显示;
- <font color="#0000ff" face="华文新魏">：设置字体的色彩并显示外观。

19.4.2 Contens 帧

在该帧中设置了一个名为“menu”的表单，在该表单中设置复选框按钮，并设置要销售的产品来形成一个目录。用户在维护该系统的时候，可以直接维护该页面的产品信息，增加一些脚本，就可以维护该页面了。这样就不会影响到其他页面的显示效果。本例设置的页面在 IE 中显示情况如图 19-21 所示。

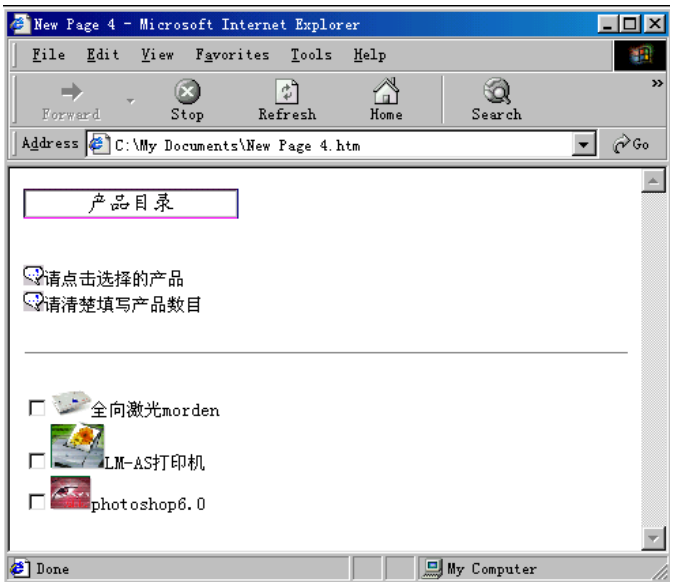


图19-21

以下为该帧的基本脚本结构：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>

<head>
<title>New Page 4</title>
<meta content="text/HTML; charset=gb2312" http-equiv="Content-Type">
<meta content="Microsoft FrontPage 3.0" name="GENERATOR">
<base target="main">
</head>

<body>
<div align="left">

<table border="1" cellPadding="0" cellSpacing="0" height="18"
style="BORDER-BOTTOM: rgb(255,0,255) 1px groove; BORDER-LEFT: rgb(0,0,128) 1px groove;
BORDER-RIGHT: rgb(0,0,128) 1px groove; BORDER-TOP: rgb(255,0,255) 1px groove"
width="158">
<TBODY>
  <tr>
    <td align="middle" height="18" width="158"><p align="center"><font face="楷体_GB2312"><big>产品目录</big></font></td>
```

```
  </tr>
</TBODY>
</table>
</div>

<p>
<br>
<img alt="mood3.gif (88 bytes)" height="15" src="mood3.gif" width="15">请点击选择的产品<br>
<img alt="mood3.gif (88 bytes)" height="15" src="mood3.gif" width="15">请清楚填写产品数目
</p>

<hr SIZE="1">

<form name="menu">
  <p>
<input name="morden" type="checkbox" value="ON">
<img alt="morden.gif (289 bytes)" src="morden.gif" width="30" height="20">全向激光 morden
<br>
  <input name="print" type="checkbox" value="print"><img
  alt="print.gif (1149 bytes)" src="print.gif" width="40" height="35">LM-AS 打印机
<br>
  <input name="photoshop" type="checkbox" value="photoshop"><img
  alt="photo6.gif (596 bytes)" src="photo6.gif" width="30" height="26">photoshop6.0
</p>
</form>
</body>
</HTML>
```

其中：

- ：设置位于该段落的元件靠左显示;
- <table border="1" cellPadding="0" cellSpacing="0" height="18"style ="BORDER-BOTTOM: rgb(255,0,255) 1px groove; BORDER-LEFT: rgb(0,0,128) 1px groove; BORDER-RIGHT: rgb(0,0,128) 1px groove; BORDER-TOP: rgb(255,0,255) 1px groove"width="158">：设置表格和表格的显示模式;
- **<hr SIZE="1">**：设置直线简单的分割窗口;
- **<form name="menu">**：设置一个名为“menu”的表单，这里设置表单的 name 属性在脚本的调用中显得尤为重要;
- **<input name="morden" type="checkbox" value="ON">**：设置一个复选按钮，在设置过程中，给出了一个 name 属性、一个 type 属性，一个 value 属性。其中，type 属性设置该表单控件的属性，即显示外观。Name 属性供脚本语言调用;
- <img alt="morden.gif (289 bytes)" src="morden.gif" width="30" height="20">全向激光 morden：设置选项实体。这里给出物品的图像，物品的名称。

19.4.3 脚 本 详 解

这里设置的 JavaScript 脚本，实现的是获得用户的选择信息，并将用户选择、订购信息传递给 main 帧的 order 表单。在设置用户输入的过程中，设置了一些过滤条件和提示信息，帮助用户完成输入过程，以免加重服务器处理的负荷，便于提高用户利用效率。

```
<script language="JavaScript">
<!--
var youjifei=0,huokuan=0,totalfei=0;

function isnumber(theinput)
{

for(var i=0;i<theinput.length;i++)
{
var isnum=theinput.substring(i,i+1);
if(isnum<"0"||isnum>"9")
{
alert("请输入数字！！！")
return false;
}
else
{
return true;
}
}
}
function choosemorden()
{
if(menu.morden.checked==true)
{
var input=prompt("请输入订购的数量");
if(input==""||input=="undefined"||input==null)
{
menu.morden.checked=false;
alert("请输入数据");
return false;
}
else
{
if(isnumber(input)==true)
{
if(confirm("单价：200 元\n 订购数量:"+input+"\n 货款合计为："+eval(200*eval(input))+"\n 邮寄费为：
"+eval(eval(200*eval(input))*0.01)))
{menu.morden.value=input;
youjifei+=eval(200*eval(input))*0.01;
huokuan+=eval(200*eval(input));
```

```
parent.main.document.order.S1.value+="\n 全向激光 morden 数量："+input+"\n";
parent.main.document.order.fei.value=youjifei;
parent.main.document.order.trade.value=huokuan;
parent.main.document.order.total.value=eval(parent.main.document.order.
trade.value)+eval(parent.main.document.order.fei.value);
return true;
}
else
{
menu.morden.checked=false;
menu.morden.value="";
}
}
}
}

}

function chooseprint()
{
if(menu.print.checked==true)
{
var inputprint=prompt("请输入订购的数量");
if(inputprint==""||inputprint=="undefined"||inputprint==null)
{
menu.print.checked=false;
alert("请输入数据");
return false;
}
else
{
if(isnumber(input)==true)
{

if(confirm("单价：5000 元\n 订购数量:"+inputprint+"\n 货款合计为：
"+eval(5000*eval(inputprint))+"\n 邮寄费为："+eval(eval(5000*eval(inputprint))*0.01)))
{menu.print.value=inputprint;
youjifei+=eval(5000*eval(inputprint))*0.01;
huokuan+=eval(5000*eval(inputprint));
parent.main.document.order.S1.value+="\nLM-AS 打印机 数量："+inputprint+"\n";
parent.main.document.order.fei.value=youjifei;
parent.main.document.order.trade.value=huokuan;
parent.main.document.order.total.value=eval(parent.main.document.order.trade.value)
+eval(parent.main.document.order.fei.value);
return true;
}
else
{
```

```
menu.print.checked=false;
menu.print.value="";
}
}
}
}

}
function choosephoto()
{
if(menu.photoshop.checked==true)
{
var inputprint=prompt("请输入订购的数量");
if(inputprint==""||inputprint=="undefined"||inputprint==null)
{
menu.photoshop.checked=false;
alert("请输入数据");
return false;
}
else
{
if(isnumber(input)==true)
{

if(confirm("单价：25 元\n 订购数量:"+inputprint+"\n 货款合计为："+eval
(25*eval(inputprint))+"\n 邮寄费为："+eval(eval(25*eval(inputprint))*0.1)))
{menu.photoshop.value=inputprint;
youjifei+=eval(25*eval(inputprint))*0.1;
huokuan+=eval(25*eval(inputprint));
parent.main.document.order.S1.value+="\nphotoshop6.0                数量："+inputprint+"\n";
parent.main.document.order.fei.value=youjifei;
parent.main.document.order.trade.value=huokuan;
parent.main.document.order.total.value=eval(parent.main.document.order.trade.value)
+eval(parent.main.document.order.fei.value);
return true;
}
else
{
menu.photoshop.checked=false;
menu.photoshop.value="";
}
}
}
}
}
// -->
</script>
```

其中：

- **var youjifei=0,huokuan=0,totalfei=0**：设置变量，并初始化设置的变量。设置这些变量的目的，就是为了有效地控制参数的传递，便于函数的调用。

1．Isnumber（）函　数

该函数用来判断用户输入的数据是否为数字。在函数的设置过程中，为了便于参数的传递和函数的调用，传递一个名为“input”的参数。

在脚本的设置过程中，设置循序，检查输入的数据，看每个字符是否为数字。在判断字符是否为数字的过程中，截取输入的字符串为单个字符。如果不为数字，则提示用户输入数字，返回一个 false 值。否则返回一个 true 值。

```
function isnumber(theinput)
{

for(var i=0;i<theinput.length;i++)
{
var isnum=theinput.substring(i,i+1);
if(isnum<"0"||isnum>"9")
{
alert("请输入数字！！！")
return false;
}
else
{
return true;
}
}
}
```

其中：

- function isnumber(theinput)：设置函数，传递参数;
- for(var i=0;i<theinput.length;i++)，i<theinput.length：获得用户输入数据的长度。该语句设置循环，控制输入数据的判断;
- **var isnum=theinput.substring(i,i+1)**：该语句利用字符串对象，将输入的数据进行截取。这里将输入的字符串截取为单个的字符，并将截取的字符赋给设置的变量;
- **if(isnum<"0"||isnum>"9")**：对截取的字符进行判断。这里判断输入数据元素是否界于 0-9 之间;
- **alert("请输入数字！！！")**：如果不为数字，则提示用户输入数据;
- **return false**：传回一个 false 值;
- **Else**：如果输入的数据是数字;
- **return true**：传回一个 true 值。

2．函数的调用

有时，为了需要，将函数尽量设置得相当小，分成不同的功能模块，然后组合这些函数，实现不同的功能。为了方便函数的调用和参数的传递，一般在设置函数的时候，就要引用相应的参数设置。

在本程序中，在应用点击选择订购的产品过程中，设置让用户输入产品的数量，然后调用该函数进行判断，看看用户输入的数据是否为数字。用户可以看看 choosemorden()函数中的设置情况。

其中：

- **if(isnumber(input)==true)**：利用该语句来判断用户的输入数据，如果用户输入的数据为数字，则返回一个 true 值，那么该语句就为真，条件成立，脚本直接向下执行。

3．Choosemorden（）函　数

利用该函数来处理当用户点击选择设置的 morden 产品时，发生的情况。

在该脚本中，先判断用户输入的情况，如果用户没有输入任何数据，则提示用户输入数据，如果用户输入的数据中没有数字或部分含有数字，则提示用户重新输入。在用户正确地输入之后，就会显示用户订购该产品的情况，然后提示用户确认该次操作。如果用户确认操作，则在第三帧中显示用户的订购情况。如果用户不确认该次操作，则取消对该对象的选择。

脚本如下：

```
function choosemorden()
{
if(menu.morden.checked==true)
{
var input=prompt("请输入订购的数量");
if(input==""||input=="undefined"||input==null)
{
menu.morden.checked=false;
alert("请输入数据");
return false;
}
else
{
if(isnumber(input)==true)
{
if(confirm("单价：200 元\n 订购数量:"+input+"\n 货款合计为："+eval(200*eval
(input))+"\n
邮寄费为："+eval(eval(200*eval(input))*0.01)))
{menu.morden.value=input;
youjifei+=eval(200*eval(input))*0.01;
huokuan+=eval(200*eval(input));
```

```
parent.main.document.order.S1.value+="\n 全向激光 morden 数量："+input+"\n";
parent.main.document.order.fei.value=youjifei;
parent.main.document.order.trade.value=huokuan;
parent.main.document.order.total.value=eval(parent.main.document.order.trade.value)+eval
(parent.main.document.order.fei.value);
return true;
}
else
{
menu.morden.checked=false;
menu.morden.value="";
}
}
}
}
}
```

其中：

- if(menu.morden.checked==true)：先判断该对象是否处于选择状态;
- var input=prompt("请输入订购的数量")：如果该产品被选择，则提示用户输入订购数量;
- if(input==""||input=="undefined"||input==null)：判断用户输入的数据。这里判断用户是否输入了数据;
- menu.morden.checked=false：如果没有输入数据，则取消对该对象的选择。
- alert("请输入数据")：提示用户输入数据;
- **return false**：返回一个 false 值，以被其他函数调用;
- **Else**：如果用户输入了数据;
- **if(isnumber(input)==true)**：调用设置的函数，判断用户输入的数据是否为数字。如果是的话，则执行如下语句;
- if(confirm("单价：200 元\n 订购数量:"+input+"\n 货款合计为："+eval(200*eval(input))+"\n 邮寄费为："+eval(eval(200*eval(input))*0.01)))：让用户确认对该产品的订购。如果用户确认该次操作，则执行如下语句;
- **{menu.morden.value=input**：将输入的数据赋给 morden，作为该对象的 value 属性值;
- youjifei+=eval(200*eval(input))*0.01：邮寄费累加;
- huokuan+=eval(200*eval(input))：货款累加;
- parent.main.document.order.S1.value+="\n 全向激光 morden 数量："+input+"\n"：将用户输入的数据和用户的选择显示在第三帧的滚动文本框中;
- parent.main.document.order.fei.value=youjifei：将邮寄费赋给 main 帧中的 order 表单的 fei 文本框，在该文本框中显示用户的邮寄费;
- **parent.main.document.order.trade.value=huokuan**：将货款总计赋给 main 帧中的 order 表单的 trade 文本框，在该文本框中显示用户的货款总计;
- parent.main.document.order.total.value=eval(parent.main.document.order.trade.

value)+eval(parent.main.document.order.fei.value);return true：自动计算用户所需的费用，这里调用 eval()方法，将用户的邮寄费和货款转换为数字，然后相加;

- **Else**：如果用户没有确认该次操作;
- menu.morden.checked=false：取消对该物品的选择;
- menu.morden.value=""：清除该物品的 value 值。

4．Onclick 事件处理器

该事件用来处理用户点击时的情况。在 JavaScript 中，利用该事件处理器处理当鼠标点击时，调用系统预定义的方法和设置的函数。

一般来说，我们在设置点击事件处理的时候，一般将该事件处理的关键词置于<input>中。

在该示例中，设置用户点击给出产品的时候，调用设置的函数，处理用户选择的产品。

```
<input name="morden" onclick="choosemorden()" type="checkbox" value="ON"><img
  alt="morden.gif (289 bytes)" src="morden.gif" width="30" height="20">全向激光 morden<br>
  <input name="print" onclick="chooseprint()" type="checkbox" value="print"><img
  alt="print.gif (1149 bytes)" src="print.gif" width="40" height="35">LM-AS 打印机<br>
  <input name="photoshop" onclick="choosephoto()" type="checkbox" value="photoshop"><img
  alt="photo6.gif (596 bytes)" src="photo6.gif" width="30" height="26">photoshop6.0
```

其中：

- <input name="morden" onclick="choosemorden()" type="checkbox" value ="ON">：当用户点击设置的 morden 产品的时候，调用设置的函数 choosemorden（）来处理相关的事件;
- <input name="print" onclick="chooseprint()" type="checkbox" value ="print">：当用户点击设置的 print 产品的时候，调用设置的函数 chooseprint（）来处理选择该产品相关的事件;
- <input name="photoshop" onclick="choosephoto()" type="checkbox" value="photoshop">：当用户点击设置的 photoshop 产品的时候，调用设置的函数 choosephoto（）来处理相关的事件。

19.4.4 Main 帧

在该帧中，设置一个名为“order”的表单来设置用户信息。在该表单中，有收集用户数据的文本框，有显示用户选购的情况。

对于用户输入的数据，这里都设置了相应的过滤条件，以减轻服务器的负担。通过 JavaScript 设置的脚本来完成数据的格式和校验。

在 IE 中浏览该帧的效果如图 19-22 所示。

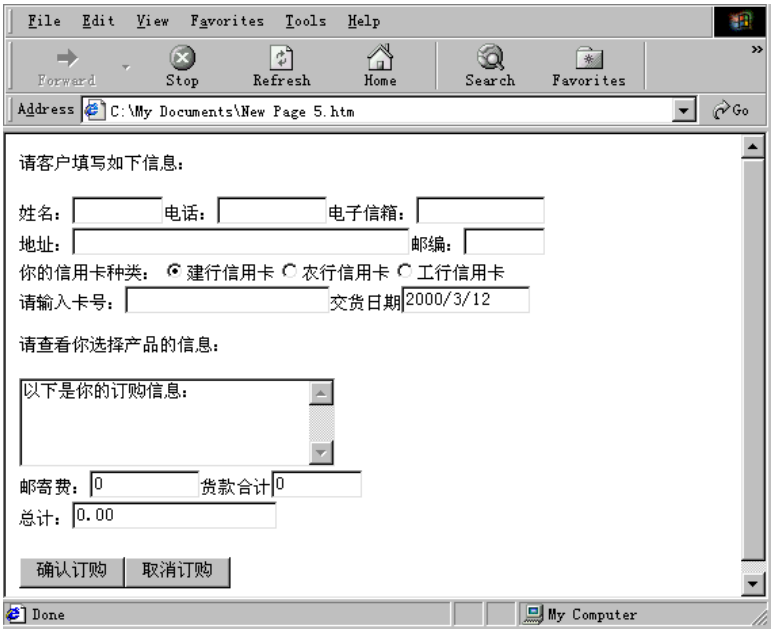


图19-22

该程序基本脚本结构如下：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>

<head>
<title>New Page 5</title>
<meta content="text/HTML; charset=gb2312" http-equiv="Content-Type">
<meta content="Microsoft FrontPage 3.0" name="GENERATOR">
<meta content="none" name="Microsoft Border">
</head>

<body>

<form action="mailto:tsg@mail.scit.com" method="post" name="order">
  <p>请客户填写如下信息：
<br>
  <br>
  姓名：<input name="name" size="8">
电话：<input  name="phone" size="10">
电子信箱：<input name="email"  size="12"><br>
  地址：<input name="adress" size="34">
邮编：<input  name="zip" size="7">
<br>
  你的信用卡种类：
<input CHECKED name="R1" type="radio" value="jianhang">建行信用卡
<input  name="R1" type="radio" value="nonghang ">农行信用卡
<input name="R1" type="radio"  value="gonghang">工行信用卡
<br>
  请输入卡号：<input name="cardnum" size="20">
```

```
交货日期<input   name="date" size="12" value="2000/3/12"><br>
  <br>
  请查看你选择的产品信息：<br>
  <br>
  <textarea cols="30" name="S1" rows="4">以下是你的订购信息：</textarea>
<br>
  邮寄费：<input name="fei" size="10" value="0">
货款合计<input name="trade" size="8"   value="0">
<br>
  总计：<input name="total" size="20" value="0.00">
</p>
  <p>
<input name="B1" type="submit" value="确认订购">
<input name="B2" type="reset" value="取消订购">
</p>
</form>
</body>
</HTML>
```

其中：

- <form action="mailto:tsg@mail.scit.com" method="post" name ="order">：设置表单，该表单的名字为 order，调用电子邮件信箱地址来处理表单信息;
- <input name="name" type=”test” size="8">：设置一个文本框，接受用户的输入或显示用户的信息;
- <input CHECKED name="R1" type="radio" value="jianhang">：设置一个单选按钮。该按钮处于预选定状态;
- <textarea cols="30" name="S1" rows="4">以下是你的订购信息：</tex- tarea>：设置一个多行文本框，在该文本框中，显示给用户四行、三十列，并且初始化信息为”以下是你的订购信息：”;
- <input name="B1" type="submit" value="确认订购">：设置一个确认按钮;
- <input name="B2" type="reset" value="取消订购">：设置一个取消按钮。

1．脚 本 详 解

在该程序中，设置了如下的脚本。这些脚本主要是验证用户的输入，引导用户输入系统要求的数据以减轻服务器的负担。

在函数的设置过程中，为了处理不同的情况，分别设置了相对于不同对象的函数，调用传递参数、嵌套函数，尽量使功能不断地完善。

```
<script language="JavaScript">
<!--
function isrequired(thefield)
{
if( thefield.value=="")
{
alert("请输入数据！！！");
```

```
return false;
}
else
{
return true;
}
}
function isnumber(theinput)
{
if(isrequired(theinput)==true)
{
for(var i=0;i<theinput.value.length;i++)
{
var isnum=theinput.value.substring(i,i+1);
if(isnum<"0"||isnum>"9")
{
alert("请输入数字！！！")
return false;
}
else
{
return true;
}
}
}
}
function isemail(theinput)
{
if(isrequired(theinput)==true)
{
var isnum=theinput.value.indexOf("@");
if(isnum!=-1&&theinput.value.indexOf(".")!=-1)
{
alert("thank！！！");
return true;
}
else
{
alert("请输入正确的邮箱地址！！");
theinput.value="";
return false;
}

                }
}
function iszip(theinput)
{
if(theinput.value.length!=6)
```

```
{
alert("邮编错误");
theinput.value="";
return false;
}
else
{
isnumber(theinput);
}
}

function isdate(theinput)
{
if(isrequired(theinput)==true)
{
var thechar=theinput.value.substring(4,5);
if(thechar!="/")
{
alert("请输入正确的日期格式(如 2000/12/25）");
return false;
}
else
{
if(confirm("交货日期为："+theinput.value+"。"))
{
alert("谢谢你，我们将及时与你联系，准时交货！！")
}
else
{
theinput.value="";
isrequired(theinput);
}
}
}
}
function quxiao()
{
parent.contents.document.menu.morden.checked=false;
parent.contents.document.menu.print.checked=false;
parent.contents.document.menu.photoshop.checked=false;
alert("请重新选取物品！");
}

function queren()
{
if(order.date.value==""||order.adress.value==""||order.name.value==""||order.tra-
de.value=="0")
{
```

```
order.action="";
alert("请填写好用户信息，选择产品，填上该产品的数量\n 确认后，
在送出该数据！ ");

return false;
}
else
{
alert("谢谢你订购本公司的产品！ ");
}
}
// -->
</script>
```

2．Isrequired（）函　数

该函数处理用户填写的数据。如果用户没有输入数据，就会提示用户输入数据。

为了便于参数的传递和函数的调用。系统设置了一个含参数的函数，并且传递相应处理的函数值。如果用户输入了数据，就传回一个真值，如果没有输入数据就传回一个 false。

该函数的脚本如下：

```
function isrequired(thefield)
{
if( thefield.value=="")
{
alert("请输入数据！ ！ ！ ");
return false;
}
else
{
return true;
}
}
```

其中：

- function isrequired(thefield)：定义函数，设置一个传递的参数;
- **if(thefield.value=="")**：判断用户的输入。这里判断用户是否输入了数据;
- **alert("请输入数据！ ！ ！ ")**：如果用户没有输入数据，则提示用户输入数据。该语句调用 window 的 alert()方法;
- **return false**：如果以上条件成立，返回一个 false 值;
- **Else**：如果用户输入了数据。该语句与上一个 if 语句相对应;
- **return true**：返回参数值 true。

3．Isnumber（）函　数

该函数判断用户输入的数据是否为数字。

在该页面中，用它来判断用户输入的数据是否为数字，主要用来处理用户输入的电话号码和信用卡号。系统简单地认为，如果用户输入的电话号码和信用卡号含有非数字字符，就会认为用户的输入出错。

该函数的脚本如下：

```
function isnumber(theinput)
{
if(isrequired(theinput)==true)
{
for(var i=0;i<theinput.value.length;i++)
{
var isnum=theinput.value.substring(i,i+1);
if(isnum<"0"||isnum>"9")
{
alert("请输入数字！！！")
return false;
}
else
{
return true;
}
}
}
}
```

其中：

- **function isnumber(theinput)**：设置函数，并传递相应的参数。这里用户可以和在 contents 帧设置的 isnumber 做一个比较，看看有什么不同。事实上，主要的一点就是这里用来处理表单的输入以及文本框中的数据;
- **if(isrequired(theinput)==true)**：调用设置的 isrequire()函数，处理用户输入的数据，判断用户是否输入了数据;
- for(var i=0;i<theinput.value.length;i++)：设置循环;
- var isnum=theinput.value.substring(i,i+1)：截取字符串为单个字符，并将获取的数据赋给设置的变量;
- if(isnum<"0"||isnum>"9")：判断截取的字符;
- **alert("请输入数字！！！")**：如果用户输入的字符中有非数字字符的存在，则提示用户;
- **return false**：返回一个 false 值，供用户调用;
- **Else**：如果输入了数据，并且为数字;
- **return true**：传回一个参数值 true，供用户调用。

4．Isemail（）函　数

该函数用来判断用户输入的邮件地址是否为有效的邮件地址。

系统在设置的过程中，只是做了一个简单的判断：我们认为如果用户输入的数据中如

果不含有“@”和“.”字符，则认为用户输入的数据非法。当然在处理该输入之前，先判断用户是否输入了数据。

```
function isemail(theinput)
{
if(isrequired(theinput)==true)
{
var isnum=theinput.value.indexOf("@");
if(isnum!=-1&&theinput.value.indexOf(".")!=-1)
{
alert("thank！！！");
return true;
}
else
{
alert("请输入正确的邮箱地址！！");
theinput.value="";
return false;
}
}
}
```

其中：

- function isemail(theinput)：设置函数，并传递调用参数;
- **if(isrequired(theinput)==true)**：调用设置的函数 isrequired()处理用户的输入，判断用户是否输入了数据;
- var isnum=theinput.value.indexOf("@")：该语句判断用户输入的字符串中是否有”@”;
- **if(isnum!=-1&&theinput.value.indexOf(".")!=-1)**：该语句先判断用户输入的数据中是否含有“.”。然后再判断用户输入的数据中是否都有“@”,”.”字符，如果两者都不含有或只含有一个，则认为用户输入的数据是非法的。否则认为用户输入了合法的数据;
- **return true**：传递一个 true 值;
- **Else**：如果用户输入了一个非法数据;
- **alert("**请输入正确的邮箱地址！！**")**：提示用户输入正确的数据;
- **return false**：传回一个 false 值。

5．Iszip（）函　数

利用该函数处理用户输入的邮编。

邮编一般为 6 位数字，先判断用户输入的数据是否为 6 位，如果不是，则提示用户输入的数据有误。如果输入的数据为 6 位，再判断用户输入的数据是否为数字。

该脚本如下：

```
function iszip(theinput)
{
```

```
if(theinput.value.length!=6)
{
alert("邮编错误");
theinput.value="";
return false;
}
else
{
isnumber(theinput);
}
}
```

其中：

- if(theinput.value.length!=6)：判断用户输入的数据是否为 6 位。如果不为 6 位数字；
- **alert("**邮编错误**")**：提示用户输入的数据有误；
- theinput.value=""：清除用户的输入；
- return false：传递一个 false 参数；
- **Else**：如果用户输入的数据为 6 位；
- **isnumber(theinput)**：调用 isnumber()函数，判断用户输入的数据是否全为数字。

6．Isdate（）函　数

该函数处理用户输入的交货日期。

交货的日期在商业的应用中是相当重要的。在该脚本中设置了简单的数据格式化，如果用户没有按规定的格式写入数据，就认为用户输入的数据非法。在数据格式的判断中，只简单地做了判断，当用户按规定的格式输入的时候，数据的第五个字符一定为“/“。如果用户输入的格式正确，则弹出一个确认对话框，让用户确认输入。

该函数的脚本如下：

```
function isdate(theinput)
{
if(isrequired(theinput)==true)
{
var thechar=theinput.value.substring(4,5);
if(thechar!="/")
{
alert("请输入正确的日期格式(如 2000/12/25）");
return false;
}
else
{
if(confirm("交货日期为："+theinput.value+"。"))
{
alert("谢谢你，我们将及时与你联系，准时交货！！")
}
else
```

```
{
theinput.value="";
isrequired(theinput);
}
}
}
}
```

其中：

- **if(isrequired(theinput)==true)**：调用 isrequired()函数，处理用户输入数据。如果用户输入了数据，就传回一个 true 值，以符合设置的判断条件;
- **var thechar=theinput.value.substring(4,5)**：调用字符串对象的 substring()方法，截取输入字符串的第五个字符，然后将该字符赋给设置的变量“thechar”；
- if(thechar!="/")：判断截取的字符是否为“/ “;
- alert("请输入正确的日期格式(如 2000/12/25)")：如果不是，提示用户输入正确的日期格式;
- **return false**：传回一个 false 值;
- **Else**：如果用户的输入正确;
- if(confirm("交货日期为："+theinput.value+"。"))：让用户确认输入的数据;
- **alert("**谢谢你，我们将及时与你联系，准时交货！！**")**：如果用户确认输入的数据，则调用 window 对象的 alert()方法提示用户注意联系;
- **Else**：如果用户不确认该交货日期;
- theinput.value=""：清除用户的输入;
- **isrequired(theinput)**：调用 isrequired()函数，让用户输入数据。

7．Onblur 事件处理器

该事件处理器用来处理表单元素失去输入焦点时候所要进行的处理。一般用该事件处理用户的输入。

在该页面中，对每个要输入的数据项目都设置了这样的事件处理器，检查用户的输入是否正确。

```
姓名：<input name="name" onblur="isrequired(order.name)" size="8">
电 话 ： <input name="phone" onblur="isnumber(order.phone)" size="10"> 电 子 信 箱 ： <input name="email" onblur="isemail(order.email)" size="12">
<br>
  地址：<input name="adress" onblur="isrequired(order.adress)" size="34">
邮编：<input name="zip" onblur="iszip(order.zip)" size="7">
<br>

  请输入卡号：<input name="cardnum" onblur="isnumber(order.cardnum)" size="20">
交货日期<input
  name="date" onblur="isdate(order.date)" size="12" value="2000/3/12">
<br>
```

其中：

- **onblur="isrequired(order.name)"**：将 order 表单中的 name 属性传递给设置的函数 isrequired()，当用户离开该字段时调用该函数处理填写的姓名字符;
- **onblur="isnumber(order.phone)"**：将 order 表单中的 phone 属性传递给设置的函数 isnumber(),当用户离开该字段时调用该函数处理填写的电话字符;
- **onblur="isemail(order.email)"**：将 order 表单中的 email 属性传递给设置的函数 isemail()，当用户离开该字段时调用该函数处理填写的电子邮件字符;
- **onblur="isrequired(order.adress)"**：将 order 表单中的 adress 属性传递给设置的函数 isrequired()，当用户离开该字段时调用该函数处理填写的地址字符;
- **onblur="iszip(order.zip)"**：将 order 表单中的 zip 属性传递给设置的函数 iszip(),当用户离开该字段时调用该函数处理填写的邮编字符;
- **onblur="isnumber(order.cardnum)" size="20">**：将 order 表单中的 cardnum 属性传递给设置的函数 isnumber()，当用户离开该字段时调用该函数处理填写的卡号字符;
- **onblur="isdate(order.date)"**：将 order 表单中的 date 属性传递给设置的函数 isdate(),当用户离开该字段时调用该函数处理填写的交货日期字符。

8．Quxiao（）函　数

该函数用来处理按下表单的取消订购按钮时的情况。

在该脚本中，当用户按下取消订购按钮的时候，就认为用户不但取消 order 表单的值,而且取消当前选择的订购产品。

```
function quxiao()
{
parent.contents.document.menu.morden.checked=false;
parent.contents.document.menu.print.checked=false;
parent.contents.document.menu.photoshop.checked=false;
alert("请重新选取物品！");
}
```

其中：

- parent.contents.document.menu.morden.checked=false：设置 contents 帧中 menu 表单的 morden 产品为非选择状态;
- parent.contents.document.menu.print.checked=false：设置 contents 帧中 menu 表单的 print 产品为非选择状态;
- parent.contents.document.menu.photoshop.checked=false：设置 contents 帧中 menu 表单的 photoshop 产品为非选择状态;
- alert("请重新选取物品！")：提示用户重新选择产品。

9．Queren（）函　数

该函数处理用户按下确认订购按钮将发生的事件。

在该脚本中，当用户按下确认按钮的时候，先判断用户的信息是否填写完毕。这里设

置了一些必填字段，比如交货日期、交货地点、客户的姓名等等。同时如果客户的货款为0，则认为用户没有选择产品，也认为这是一个非法交易。凡此种种，都认为为非法操作，就会取消用户的输入信息，提示用户输入完全。否则，则提示用户当前交易完成，谢谢惠顾。

该脚本如下：

```
function queren()
{
if(order.date.value==""||order.adress.value==""||order.name.value==""
||order.trade.value=="0")
{
order.action="";
alert("请填写好用户信息，选择产品，填上该产品的数量\n 确认后，
在送出该数据！");

return false;
}
else
{
alert("谢谢你订购本公司的产品！");
}
}
```

其中：

- if(order.date.value==""||order.adress.value==""||order.name.value==""||order.trade.value=="0"): 设置判断条件。这里主要过滤如下条件：客户的订购情况，如果客户的货款为 0，则认为用户没有选择产品，无诚意进行交易。客户的交货日期、客户的交货地点、客户的姓名等等，这些都是要求必须填写的;
- **order.action=""**：设置 order 表单的 action 属性为空。这样就不会调用设置的表单发送数据，并控制表单的动作;
- alert("请填写好用户信息，选择产品，填上该产品的数量\n 确认后，再送出该数据！");
- 如果用户没有通过设置的过滤条件，这时提示用户进行正确的操作;
- **return false**：传递一个 false 值;
- **Else**：如果通过了系统设置的过滤条件。这时系统就会调用设置的信箱动作处理表单的输入，将收集到的客户信息传递给邮件的拥有者;
- alert("谢谢你订购本公司的产品！"): 提示信息，感谢用户的惠顾。

10．Onclick 事件处理器

该事件处理器用来处理链接、按钮、图标按钮、提交按钮、复选框、单选按钮、复位按钮被点击时的情况。一个点击可以开启链接的页面，调用系统设置的函数。

在该脚本中，利用 onclick 事件处理器来处理用户按下订购按钮、取消按钮时，调用设置的 quxiao()函数、queren()函数。

```
<input name="B1" type="submit" value="确认订购"
onclick="queren()">
<input name="B2"
   onclick="quxiao()" type="reset" value="取消订购">
```

其中：

- **onclick="queren()"**：设置用户按下确认订购按钮时，调用设置的 queren()函数，处理用户信息;
- **onclick="quxiao()"**：设置用户按下取消订购按钮时，调用设置的 quxiao()函数，清除用户信息。

第20章

2000 珍藏版

主要内容

- cookie 入　门
- cookie 实　例

本章导读

本章主要通过 JavaScript 构造了一个人事查询系统，避开复杂的数据库设计，利用脚本语言实现数据查询。通过该例的学习，用户可以不用数据库的操作，轻易地实现远程查询，可以自如地设置自己的搜索工具，创建本地的搜索。

20.57 Cookie 入 门

Netscape 设置了 cookie，用来设置保存计算机上的信息，在计算机关闭的时候，该 cookie 信息释放。

Cookies 是存储在客户机上的小数据包，以后的服务器都可以访问放置该 cookies 数据包的客户机。Cookie 由响应浏览器的请求服务程序发送的信息组成的。根据 CGI 程序的 URL，将发送的 cookie 信息存放在本机的 cookies.txt 文件中。这个 URL 的格式，可以随着客户机上的其他响应信息而统一化。

供 CGI 使用的 cookie，提供了向浏览器保存信息的功能。浏览器请求 CGI 程序的 URL 时，将这个信息返回给 CGI 程序；同时，CGI 响应浏览器的请求，更新系统的 cookie。

脚本程序中都有一个 set-cookie 信息头的部分，作为 http 响应创建 cookie.cookie 信息存放的方法如下：

set-cookie :name=values;expires=date;path=PATH;domain=domain_name;secure.

其中：

- **name=values**：是必须的，其他语句是可以选择的;
- **Name=values**：这是 cookie 的 name 属性。在设置的时候应给定一个值。如：email=”tsg@mail.scit.cn.”。在设置的字符中不允许有逗号、分号、空格符，程序员可以对该属性值进行编码;
- Expires=DATE：该语句设置 cookie 的过期时间。如果用户没有指定的话，则关闭浏览器的时候该 cookie 设置失效;

在该 DATE 格式中，须按如下的格式指定日期：

星期，月-日-年 时:分:秒 GMT

如：

Friday,20-9-99 12 11:45:15 GMT

GMT 指格林尼治时间。

Domain=domain_name，该语句指定存放 url 请求的计算机域名。设置的格式如下：

domain=scit.com.cn

- **Path=PATH**：设置与 cookie 文件相关的路径。这里的路径指的是广泛路径。如：http://202.115.15.2/cgi/1.asp，在设置 path 的时候，应指定为 path=/cgi 的形式;
- **Secure**：如果用户指定了该语句，那么 cookie 通过 http 服务器发送用户的请求。

20.58 实 例 特 性

在本章设置示例来讲解 cookie 的设置和检索方法。通过本例的学习，用户可以设置

cookie 文件，检索 cookie，以及显示用户设置和选择的 cookie 文件。

该示例的特性如下：

- Cookie 信息存放的方法;
- Cookie 文件的设置方法;
- Cookie 文件的存储;
- Cookie 文件的检索;
- Cookie 文件的清除;
- 利用 JavaScript 动态生成页面;
- 利用 JavaScript 数组设置数据表。

20.59 程序源代码

实现如上特性的源代码如下：

```
<html>

<head><script language="JavaScript">
<!--

//显示存在的 cookie.
function GetCookie(name)
{
   var result = null;
   var myCookie = " " + document.cookie + ";";
   var searchName = " " + name + "=";
   var startOfCookie = myCookie.indexOf(searchName);
   var endOfCookie;
   if (startOfCookie != -1)
   {
      startOfCookie += searchName.length;
      // skip past cookie name
      endOfCookie = myCookie.indexOf(";", startOfCookie);
      result = unescape(myCookie.substring(startOfCookie,endOfCookie));
   }
   return result;
}

//设置 cookie.
function SetCookieEZ(name, value)
{
   document.cookie = name + "=" + escape(value);
}
```

```
function SetCookie(name, value, expires, path, domain, secure)
{
  var expString = ((expires == null)? "" : ("; expires=" + expires.toGMTString()));
  var pathString = ((path == null) ? "" : ("; path=" + path));
  var domainString = ((domain == null)? "" : ("; domain=" + domain));
  var secureString = ((secure == true) ? "; secure" : "");
  document.cookie = name + "=" + escape(value)+ expString + pathString + domainString+
secureString;
}

//清除 cookie.
function ClearCookie(name)
{
  var ThreeDays = 3 * 24 * 60 * 60 * 1000;
  var expDate = new Date();
  expDate.setTime (expDate.getTime() - ThreeDays);
  document.cookie = name + "=ImOutOfHere; expires="+ expDate.toGMTString();
}

//cookie 属性表。
/* Here is our "favorite" object.
   Properties: fullName - The full descriptive name
               cook     - The code used for the cookie
               urlpath  - The full url (http://...) to the site
      Methods: Enabled - Returns true if the link's cookie is
                         turned on
               Checked  - Returns the word "CHECKED" if the
                          link's cookie is turned on
               WriteAsCheckBox - Sends text to the document in a
                                 checkbox control format
               WriteAsWebLink  - Sends text to the document in a
                                 <a href...> format
-----------------------------------------------------------------*/
function favorite(fullName, cook, urlpath)
{
  this.fullName = fullName;
  this.cook     = cook;
  this.urlpath  = urlpath;
  this.Enabled = Enabled;
  this.Checked = Checked;
  this.WriteAsCheckBox = WriteAsCheckBox;
  this.WriteAsWebLink = WriteAsWebLink;
}

//探测 cookie 的存在。
function Enabled()
{
  var result = false;
```

```
        var FaveCookie = GetCookie("Favorites");
        if (FaveCookie != null)
        {
          var searchFor = "<" + this.cook + ">";
          var startOfCookie = FaveCookie.indexOf(searchFor);
          if (startOfCookie != -1)
            result = true;
        }
        return result;
      }

      function Checked ()
      {
        if (this.Enabled())
          return "CHECKED ";
        return "";
      }

       //设置 cookie.
      function WriteAsCheckBox ()
      {
        // Check to see if it's a title or regular link
        if (this.urlpath == "")
        {
          // It's a section title
          result = '<dt><strong>' + this.fullName + '</strong>';
        }
        else
        {
          // It's a regular link
          result = '<dd><input type="checkbox" name="'+ this.cook + '" '+ this.Checked()+
'onClick="SetFavoriteEnabled(this.name, this.checked);">'+ this.fullName;
        }
        document.write(result);
      }

      var NextHeading = "";

      function WriteAsWebLink()
      {
        var result = '';
        if (this.urlpath == "")
        {
          NextHeading = this.fullName;      // It's must be a Title
        }
        else
        {
          if (this.Enabled() || (GetCookie("ViewAll") == "T"))
```

```
            {
              if (NextHeading != "")
              {
                result = '<p><dt><strong>' + NextHeading+ '</strong>';
                NextHeading = "";
              }
              result = result + '<dd><a href="' + this.urlpath + '">'+ this.fullName + '</a>';
            }
          }
          document.write(result);
        }


        var FaveList = new Array();
        // Comics Section --------------------
        FaveList[1] = new favorite("Comics", "", "");
        FaveList[2] = new favorite("Dilbert", "cdilb","http://www.unitedmedia.com/comics/dilbert/");
        FaveList[3] = new favorite("Doonesbury", "cdoon","http://www.uexpress.com/cgi-
bin/ups/mainindex.cgi?code=db");
        FaveList[4] = new favorite("Mr. Boffo", "cboff","http://www.uexpress.com/cgi-
bin/ups/new_mainindex.cgi?code=mb");
        // General News Section -------------
        FaveList[5] = new favorite("General News", "", "");
        FaveList[6] = new favorite("CNN", "ncnn", "http://www.cnn.com/");
        FaveList[7] = new favorite("NPR", "nnpr","http://www.npr.org/news/");
        FaveList[8] = new favorite("Boston Globe", "nbos","http://www.boston.com/");
        // Computer Industry Section --------
        FaveList[9] = new favorite("Computer Industry", "", "");
        FaveList[10] = new favorite("PC Week", "ipcw","http://www.pcweek.com/");
        FaveList[11] = new favorite("TechWeb", "icmp","http://www.techweb.com/wire/wire.html");
        FaveList[12] = new favorite("Netscape", "ntsc","http://devedge.netscape.com/");
        FaveList[13] = new favorite("Microsoft", "micr","http://msdn.microsoft.com/");
        // Search Engines Section -----------
        FaveList[14] = new favorite("Search Engines", "", "");
        FaveList[15] = new favorite("Yahoo!", "syah","http://www.yahoo.com/");
        FaveList[16] = new favorite("Alta Vista", "sav","http://www.altavista.com/");
        FaveList[17] = new favorite("Excite", "sexc","http://www.excite.com/");
        // Auction Section ------------------
        FaveList[18] = new favorite("Auctions", "", "");
        FaveList[19] = new favorite("ebay", "ebay","http://www.ebay.com/");
        FaveList[20] = new favorite("Yahoo Auctions", "yhac","http://auctions.yahoo.com/");
        // Misc. Section --------------------
        FaveList[21] = new favorite("Misc.", "", "");
        FaveList[22] = new favorite("Today in History", "mtih","http://www.thehistorynet.com/today/
today.htm");
        FaveList[23] = new favorite("Merriam-Webster's Word of the Day","mwod", "http://www.m-w.com/
cgi-bin/mwwod.pl");
        FaveList[24] = new favorite("Quotes of the Day", "mquot","http://www.starlingtech.com/quotes/
```

```
qotd.html");

    //用户选择。
    function SendOptionsPage()
    {
      document.write('<h1>在以下选择，并作为你的珍藏</h1>');
      document.write('<form method=post>');
      // Here's the button for viewing the Favorites page
      document.write('<input type=button value="显示选择的内容" '+ 'onClick="'+'ReloadPage()'+';">');
      // The links will look nicer inside a definition listdocument.write('<dl>');
      for (var i = 1; i < FaveList.length; i++)
        FaveList[i].WriteAsCheckBox();
      // Write each checkbox
      document.write('</dl><p>');
      ClearCookie("ViewAll");
      document.write('</form>');
    }

    //显示。
    function LoadOptions()
    {
      SetCookieEZ("ShowOptions", "T");
      window.open(document.location.href, "_top", "");
    }


    function ToggleView()
    {
      if (GetCookie("ViewAll") == "T")
      {
        ClearCookie("ViewAll");
      }
      else
      {
        var fiveYears = 5 * 365 * 24 * 60 * 60 * 1000;
        var expDate = new Date();
        expDate.setTime (expDate.getTime() + fiveYears );
        SetCookie("ViewAll", "T", expDate, null, null, false);
      }
      window.open(document.location.href, "_top", "");
    }

    //显示用户的链接。
    function SendPersonalPage()
    {
      if (GetCookie("ViewAll") != "T")
        document.write('<h1>你的珍藏:</h1>');
      else
```

```
            document.write('<h1>我的链接:</h1>');
        // Here are the buttons for viewing the options or
        // "View All" pages
        document.write('<form method=post>');
        if (GetCookie("ViewAll") == "T")
        {
            document.write('<input type=button value="View Favorites" '+'onClick="ToggleView();">');
        }
        else
        {
            document.write('<input type=button value="显示我的珍藏" '+'onClick="ToggleView();">');
        }
        document.write('<input  type=button  '+  'value="点击选择我的珍藏"  '+  'onClick="LoadOptions();
">');
        document.write('</form>');
        // The links will look nicer inside a definition list
        document.write('<dl>');
        for (var i = 1; i < FaveList.length; i++)
            FaveList[i].WriteAsWebLink();        // Write each link
        document.write('</dl><p>');
    }

    //-------------------------------------------
    function isEnabled(name)
    {
        var result = false;
        var FaveCookie = GetCookie("Favorites");
        if (FaveCookie != null)
        {
            var searchFor = "<" + name + ">";
            var startOfCookie = FaveCookie.indexOf(searchFor);
            if (startOfCookie != -1)
                result = true;
        }
        return result;
    }

    //---------------------------------------------------------------
    //增加数据。
    function AddFavorite(name)
    {
        if (!isEnabled(name))
        {
            var fiveYears = 5 * 365 * 24 * 60 * 60 * 1000;
            var expDate = new Date();
            expDate.setTime (expDate.getTime() + fiveYears );
            SetCookie("Favorites", GetCookie("Favorites")+ "<" + name + ">", expDate, null, null, false);
        }
```

```
}

//-----------------------------------------------------------------
//清除用户的设置。
function ClearFavorite(name)
{
  if (isEnabled(name))
  {
    var FaveCookie = GetCookie("Favorites");
    var searchFor = "<" + name + ">";
    var startOfCookie = FaveCookie.indexOf(searchFor);
    var NewFaves = FaveCookie.substring(0, startOfCookie)+ FaveCookie.substring(startOfCookie+
searchFor.length,FaveCookie.length);
    var fiveYears = 5 * 365 * 24 * 60 * 60 * 1000;
    var expDate = new Date();
    expDate.setTime (expDate.getTime() + fiveYears );
    SetCookie("Favorites", NewFaves, expDate, null, null, false);
  }
}

//-----------------------------------------------------------------
// 用户设置。

function SetFavoriteEnabled(name, SetOn)
{
  if (SetOn)
    AddFavorite(name);
  else
    ClearFavorite(name);
}

//-----------------------------------------------------------------
//重新载入页面。
//-----------------------------------------------------------------
function ReloadPage()
{
  window.open(document.location.href, "_top", "");
}

// End Commented Script -->
</script>

<title></title>
</head>

<body>
<script language="JavaScript">
<!--
```

```
if (GetCookie("ShowOptions") == "T")
{
   ClearCookie("ShowOptions");
   SendOptionsPage();
}
else
{
   SendPersonalPage();
}
// End Commented Script -->
</script>

<p align="center"><font face="楷体_GB2312" color="#0000FF"><big>cookies 文件的设置.<br>
</big></font></p>
</body>
</html>
```

20.60 功能概述

该程序在 IE 中浏览的情况如图 20-1 所示。

图20-1

在该页面中设置了两个按钮，如果用户点击第一个按钮，就会显示用户收集的链接站点。如图 20-2 所示。

图20-2

点击选择按钮，则出现选择页面信息。在该页面的项目前面有一个复选框，供用户选择。如图 20-3 所示。

图20-3

按下显示选择内容按钮后，就会出现如图 20-4 所示的页面信息。

图20-4

点击选择显示我的珍藏按钮，就会出现我的链接页面。该页面的显示情况如图 20-5

所示。

图20-5

20.61 程序详解

20.5.1 基本脚本结构

该程序主要是设置 cookie 文件，一般更多的是利用脚本来实现。基本的脚本结构如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>New Page 2</title>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
</head>
<body>
<p><font color="#0000ff" face="楷体_GB2312"><big>cookies 文件的设置.<br>
</big></font></p>
</body>
</html>
```

在该页面中，简单地设置了一句话，表示这里设置的 cookies 文件。其中，<p>设置段落。<font color="#0000ff" face="楷体_GB2312">，设置字体的颜色和字体。<big>设置字体加大显示。

20.5.2 脚 本 详 解

在该程序中，主要是利用脚本来实现的，在设置的脚本中，尽量将 cookie 文件的设置模块化，希望通过该例的学习，深入的掌握 cookie 文件的属性、方法。同时也逐渐巩固前面所学的东西，通过这些脚本的设置，希望用户掌握 JavaScript 方面的高级编程的知识。该程序的脚本设置如下：

```
<script language="JavaScript">
<!--

//显示存在的 cookie.
function GetCookie(name)
{
   var result = null;
   var myCookie = " " + document.cookie + ";";
   var searchName = " " + name + "=";
   var startOfCookie = myCookie.indexOf(searchName);
   var endOfCookie;
   if (startOfCookie != -1)
   {
      startOfCookie += searchName.length;
      // skip past cookie name
      endOfCookie = myCookie.indexOf(";", startOfCookie);
      result = unescape(myCookie.substring(startOfCookie,endOfCookie));
   }
   return result;
}

//设置 cookie.
function SetCookieEZ(name, value)
{
   document.cookie = name + "=" + escape(value);
}


function SetCookie(name, value, expires, path, domain, secure)
{
   var expString = ((expires == null)? "" : ("; expires=" + expires.toGMTString()));
   var pathString = ((path == null) ? "" : ("; path=" + path));
   var domainString = ((domain == null)? "" : ("; domain=" + domain));
   var secureString = ((secure == true) ? "; secure" : "");
   document.cookie = name + "=" + escape(value)+ expString + pathString + domainString+
secureString;
}

//清除 cookie.
function ClearCookie(name)
```

```
{
   var ThreeDays = 3 * 24 * 60 * 60 * 1000;
   var expDate = new Date();
   expDate.setTime (expDate.getTime() - ThreeDays);
   document.cookie = name + "=ImOutOfHere; expires="+ expDate.toGMTString();
}

//cookie 属性表
/* Here is our "favorite" object.
     Properties: fullName - The full descriptive name
                 cook       - The code used for the cookie
                 urlpath    - The full url (http://...) to the site
         Methods: Enabled - Returns true if the link's cookie is
                            turned on
                  Checked    - Returns the word "CHECKED" if the
                               link's cookie is turned on
                  WriteAsCheckBox - Sends text to the document in a
                                    checkbox control format
                  WriteAsWebLink    - Sends text to the document in a
                                      <a href...> format
-----------------------------------------------------------------*/
function favorite(fullName, cook, urlpath)
{
   this.fullName = fullName;
   this.cook     = cook;
   this.urlpath    = urlpath;
   this.Enabled = Enabled;
   this.Checked = Checked;
   this.WriteAsCheckBox = WriteAsCheckBox;
   this.WriteAsWebLink = WriteAsWebLink;
}

//探测 cookie 的存在
function Enabled()
{
   var result = false;
   var FaveCookie = GetCookie("Favorites");
   if (FaveCookie != null)
   {
      var searchFor = "<" + this.cook + ">";
      var startOfCookie = FaveCookie.indexOf(searchFor);
      if (startOfCookie != -1)
         result = true;
   }
   return result;
}

function Checked ()
```

```
{
   if (this.Enabled())
      return "CHECKED ";
   return "";
}

 //设置 cookie.
function WriteAsCheckBox ()
{
   // Check to see if it's a title or regular link
   if (this.urlpath == "")
   {
      // It's a section title
      result = '<dt><strong>' + this.fullName + '</strong>';
   }
   else
   {
      // It's a regular link
      result = '<dd><input type="checkbox" name="'+ this.cook + '" '+ this.Checked()+
'onClick="SetFavoriteEnabled(this.name, this.checked);">'+ this.fullName;
   }
   document.write(result);
}

var NextHeading = "";

function WriteAsWebLink()
{
   var result = '';
   if (this.urlpath == "")
   {
      NextHeading = this.fullName;      // It's must be a Title
   }
   else
   {
      if (this.Enabled() || (GetCookie("ViewAll") == "T"))
      {
         if (NextHeading != "")
         {
            result = '<p><dt><strong>' + NextHeading+ '</strong>';
            NextHeading = "";
         }
         result = result + '<dd><a href="' + this.urlpath + '">'+ this.fullName + '</a>';
      }
   }
   document.write(result);
}
```

```
var FaveList = new Array();
// Comics Section --------------------
FaveList[1] = new favorite("Comics", "", "");
FaveList[2] = new favorite("Dilbert", "cdilb","http://www.unitedmedia.com/comics/dilbert/");
FaveList[3] = new favorite("Doonesbury", "cdoon","http://www.uexpress.com/cgi-
bin/ups/mainindex.cgi?code=db");
FaveList[4] = new favorite("Mr. Boffo", "cboff","http://www.uexpress.com/cgi-
bin/ups/new_mainindex.cgi?code=mb");
// General News Section -------------
FaveList[5] = new favorite("General News", "", "");
FaveList[6] = new favorite("CNN", "ncnn", "http://www.cnn.com/");
FaveList[7] = new favorite("NPR", "nnpr","http://www.npr.org/news/");
FaveList[8] = new favorite("Boston Globe", "nbos","http://www.boston.com/");
// Computer Industry Section --------
FaveList[9] = new favorite("Computer Industry", "", "");
FaveList[10] = new favorite("PC Week", "ipcw","http://www.pcweek.com/");
FaveList[11] = new favorite("TechWeb", "icmp","http://www.techweb.com/wire/wire.html");
FaveList[12] = new favorite("Netscape", "ntsc","http://devedge.netscape.com/");
FaveList[13] = new favorite("Microsoft", "micr","http://msdn.microsoft.com/");
// Search Engines Section -----------
FaveList[14] = new favorite("Search Engines", "", "");
FaveList[15] = new favorite("Yahoo!", "syah","http://www.yahoo.com/");
FaveList[16] = new favorite("Alta Vista", "sav","http://www.altavista.com/");
FaveList[17] = new favorite("Excite", "sexc","http://www.excite.com/");
// Auction Section ------------------
FaveList[18] = new favorite("Auctions", "", "");
FaveList[19] = new favorite("ebay", "ebay","http://www.ebay.com/");
FaveList[20] = new favorite("Yahoo Auctions", "yhac","http://auctions.yahoo.com/");
// Misc. Section --------------------
FaveList[21] = new favorite("Misc.", "", "");
FaveList[22] = new favorite("Today in History",
"mtih","http://www.thehistorynet.com/today/today.htm");
FaveList[23] = new favorite("Merriam-Webster's Word of the Day","mwod", "http://www.m-
w.com/cgi-bin/mwwod.pl");
FaveList[24] = new favorite("Quotes of the Day",
"mquot","http://www.starlingtech.com/quotes/qotd.html");

//=====================================================================
// Page Writing Routines
//=====================================================================
//---------------------------------------------------------------
// SendOptionsPage - Writes a page allowing the user to select
//                           her favorite preferences
//---------------------------------------------------------------
function SendOptionsPage()
{
   document.write('<h1>在以下选择并作为你的珍藏</h1>');
```

```
   document.write('<form method=post>');
   // Here's the button for viewing the Favorites page
   document.write('<input type=button value="显示选择的内容" '+
'onClick="'+'ReloadPage()'+';">');
   // The links will look nicer inside a definition listdocument.write('<dl>');
   for (var i = 1; i < FaveList.length; i++)
      FaveList[i].WriteAsCheckBox();
   // Write each checkbox
   document.write('</dl><p>');
   ClearCookie("ViewAll");
   document.write('</form>');
}

//---------------------------------------------------------------
// LoadOptions - Sets the ShowOptions cookie, which makes the
//                        option selection page appear when the page is
//                        then reloaded.
//---------------------------------------------------------------
function LoadOptions()
{
   SetCookieEZ("ShowOptions", "T");
   window.open(document.location.href, "_top", "");
}

function ToggleView()
{
   if (GetCookie("ViewAll") == "T")
   {
      ClearCookie("ViewAll");
   }
   else
   {
      var fiveYears = 5 * 365 * 24 * 60 * 60 * 1000;
      var expDate = new Date();
      expDate.setTime (expDate.getTime() + fiveYears );
      SetCookie("ViewAll", "T", expDate, null, null, false);
   }
   window.open(document.location.href, "_top", "");
}

//---------------------------------------------------------------
// SendPersonalPage - Writes a page showing the categories and
//                              links which the user prefers. Only shows a
//                              heading if one of its favorites is enabled
//---------------------------------------------------------------
function SendPersonalPage()
{
   if (GetCookie("ViewAll") != "T")
```

```
      document.write('<h1>你的珍藏:</h1>');
   else
      document.write('<h1>我的链接:</h1>');
   // Here are the buttons for viewing the options or
   // "View All" pages
   document.write('<form method=post>');
   if (GetCookie("ViewAll") == "T")
   {
      document.write('<input type=button value="显示" '+'onClick="ToggleView();">');
   }
   else
   {
      document.write('<input type=button '+ ' value="显示的珍藏" '+'onClick="ToggleView();">');
   }
   document.write('<input type=button '+ 'value="点击选择显示我的珍藏： " '+
'onClick="LoadOptions();">');
   document.write('</form>');
   // The links will look nicer inside a definition list
   document.write('<dl>');
   for (var i = 1; i < FaveList.length; i++)
      FaveList[i].WriteAsWebLink();        // Write each link
   document.write('</dl><p>');
}

//====================================================================
// Helper Functions
//====================================================================
//--------------------------------------------------------------
// isEnabled - Returns True if the favorite identified by the
//                      name parameter is enabled.
//--------------------------------------------------------------
function isEnabled(name)
{
   var result = false;
   var FaveCookie = GetCookie("Favorites");
   if (FaveCookie != null)
   {
      var searchFor = "<" + name + ">";
      var startOfCookie = FaveCookie.indexOf(searchFor);
      if (startOfCookie != -1)
         result = true;
   }
   return result;
}

function AddFavorite(name)
{
   if (!isEnabled(name))
```

```
    {
        var fiveYears = 5 * 365 * 24 * 60 * 60 * 1000;
        var expDate = new Date();
        expDate.setTime (expDate.getTime() + fiveYears );
        SetCookie("Favorites", GetCookie("Favorites")+ "<" + name + ">", expDate, null, null, false);
    }
}

function ClearFavorite(name)
{
    if (isEnabled(name))
    {
        var FaveCookie = GetCookie("Favorites");
        var searchFor = "<" + name + ">";
        var startOfCookie = FaveCookie.indexOf(searchFor);
        var NewFaves = FaveCookie.substring(0, startOfCookie)+
FaveCookie.substring(startOfCookie+searchFor.length,FaveCookie.length);
        var fiveYears = 5 * 365 * 24 * 60 * 60 * 1000;
        var expDate = new Date();
        expDate.setTime (expDate.getTime() + fiveYears );
        SetCookie("Favorites", NewFaves, expDate, null, null, false);
    }
}


function SetFavoriteEnabled(name, SetOn)
{
    if (SetOn)
        AddFavorite(name);
    else
        ClearFavorite(name);
}

function ReloadPage()
{
    window.open(document.location.href, "_top", "");
}

// End Commented Script -->
</script>


<script language="JavaScript">
<!--
if (GetCookie("ShowOptions") == "T")
{
    ClearCookie("ShowOptions");
    SendOptionsPage();
```

```
}
else
{
   SendPersonalPage();
}
// End Commented Script -->
</script>
```

其中各个函数介绍如下：

1. getcookie（）函　数

该函数用来获得要检索的 cookie 值。在该脚本中，设置了一系列的变量来截取 cookie 值。最后，返回一个函数值来供其他的函数脚本调用。

为了方便函数的调用，在设置的函数中，设置了一个 name 参数。

该函数的脚本如下：

```
function GetCookie(name)
{
   var result = null;
   var myCookie = " " + document.cookie + ";";
   var searchName = " " + name + "=";
   var startOfCookie = myCookie.indexOf(searchName);
   var endOfCookie;
   if (startOfCookie != -1)
   {
      startOfCookie += searchName.length;
          endOfCookie = myCookie.indexOf(";", startOfCookie);
      result = unescape(myCookie.substring(startOfCookie,endOfCookie));
   }
   return result;
}
```

其中：

- **function GetCookie(name)**：设置函数，传递一个 name 的 arg 参数;
- **var result = null**：设置变量，设置变量的值为空;
- **var myCookie = " " + document.cookie + ";"**：取得文档的 cookie 值，并将它赋给设置的参数 mycookie;
- **var searchName = " " + name + "="**：设置检索参数;
- **var startOfCookie = myCookie.indexOf(searchName)**：调用字符串对象的 indexof()函数在设置的 cookie 对象中查看检索对象是否存在。如果该对象存在，就返回该字符串在 cookie 文件中的位置值，如果不存在则返回一个-1 值。最后将检索到的值赋给设置的变量;
- **var endOfCookie**：设置变量;
- **if (startOfCookie != -1)**：如果不存在检索对象;
- **startOfCookie += searchName.length**：重新设置检索函数值。这里设置的是检索字段的起始位置;

- **endOfCookie = myCookie.indexOf(";", startOfCookie)**：设置检索值的最后一个字符的位置值;
- **result= unescape(myCookie.substring(startOfCookie,endOfCookie))**：unescape（）为 JavaScript 的内置函数，它返回一个基于 ASCII 码的字符。这里在 cookie 字段中截取获得检索的字符串，并将该字符串传递给设置的变量;
- **return result**：返回设置的变量，供其他函数调用。

2. SetcookieEZ（）函　数

该函数用来设置文档的 cookie。由前面的内容可知，cookie 值的设置都是属性=值的情况。该脚本设置用户的 cookie 值。其脚本如下：

```
function SetCookieEZ(name, value)
{
   document.cookie = name + "=" + escape(value);
}
```

其中：

- **document.cookie = name + "=" + escape(value)**：设置 cookie 的 name 属性值。

3. Setcookie()函　数

该函数用来设置 cookie 的属性值。在该脚本中，都设置了一些判断，如果用户没有设置相应的 cookie 属性值，则该项设置为空。然后将用户设置的 cookie 赋给用户当前的文档，该脚本如下：

```
function SetCookie(name, value, expires, path, domain, secure)
{
   var expString = ((expires == null)? "" : ("; expires=" + expires.toGMTString()));
   var pathString = ((path == null) ? "" : ("; path=" + path));
   var domainString = ((domain == null)? "" : ("; domain=" + domain));
   var secureString = ((secure == true) ? "; secure" : "");
   document.cookie = name + "=" + escape(value)+ expString + pathString + domainString+
secureString;
}
```

其中：

- **var expString = ((expires == null)? "" : ("; expires=" + expires.toGMTString()))**：设置用户 cookie 的过期时间，如果用户没有设置该项目，就设置过期时间为空，否则，调用时间对象的 togmtstring()方法，将用户的设置转换为规定的格林尼治时间;
- **var pathString = ((path == null) ? "" : ("; path=" + path))**：设置 cookie 的 path 值;
- **var domainString = ((domain == null)? "" : ("; domain=" + domain))**：设置用户 URL 请求的主机域名;

- **var secureString = ((secure == true) ? "; secure" : "")**：设置安全属性;
- **document.cookie = name + "=" + escape(value)+ expString + pathString + domainString+ secureString**：将用户的设置赋给用户当前文档，作为文档对象 cookie 属性的值。

4. favorite（）函　数

该函数设置用户 favorite 对象。该函数设置一个新的对象并设置其属性、属性的值。这些属性由以下部分组成：fullname、cook、urlpath、enable、checked、writeaschecked、writeasweblink。

```
Function favorite(fullName, cook, urlpath)
{
   this.fullName = fullName;
   this.cook      = cook;
   this.urlpath    = urlpath;
   this.Enabled = Enabled;
   this.Checked = Checked;
   this.WriteAsCheckBox = WriteAsCheckBox;
   this.WriteAsWebLink = WriteAsWebLink;
}
```

5. Enable()函　数

该函数检查用户设置的 cookie 值中是否存在要检索的字段，如果存在则返回一个 true 值，否则传回一个 false 值。该函数同时作为 favorite 对象的方法，供该对象直接调用。

在该函数设置的过程中，调用设置的 getcookie（）函数，获得用户检索字段。

```
function Enabled()
{
   var result = false;
   var FaveCookie = GetCookie("Favorites");
   if (FaveCookie != null)
   {
      var searchFor = "<" + this.cook + ">";
      var startOfCookie = FaveCookie.indexOf(searchFor);
      if (startOfCookie != -1)
         result = true;
   }
   return result;
}
```

其中：

- **var result = false**：设置一个逻辑变量，并设置该变量的初始值为 false;
- **var FaveCookie = GetCookie("Favorites")**：调用设置的 getcookie（）函数，获得“favorite”检索的字段位置;

- **if (FaveCookie != null)**：如果该对象存在；
- **var searchFor = "<" + this.cook + ">"**：设置检索字段；
- **var startOfCookie = FaveCookie.indexOf(searchFor)**：获得检索字段的起始位置；
- **if (startOfCookie != -1)**：如果该检索字段在 cookie 中存在；
- **result = true**：返回一个 true 值；
- **return result**：返回参数值。

6. Writeascheckbox（）函　数

设置该函数供用户选择要显示的内容。如果用户要显示该 cookie 值，就点击设置值前的复选框。

该脚本的设置调用 document 对象的 write（）方法，显示系统设置的对象实体。其脚本如下：

```
function WriteAsCheckBox ()
{

  if (this.urlpath == "")
  {

    result = '<dt><strong>' + this.fullName + '</strong>';
  }
  else
  {

    result = '<dd><input type="checkbox" name="'+ this.cook + '" '+ this.Checked()+
'onClick="SetFavoriteEnabled(this.name, this.checked);">'+ this.fullName;
  }
  document.write(result);
}
```

其中：

- **if (this.urlpath == "")**：如果用户设置的 cookie 值为空。这里利用 path 值为空的属性，显示用户的分类标题；
- **result = '<dt>' + this.fullName + ''**：加粗显示设置的值。这里加粗显示用户的分类标题；
- **else**：如果 path 值不为空；
- **result = '<dd><input type="checkbox" name="'+ this.cook + '" '+ this.Checked()+ 'onClick="SetFavoriteEnabled(this.name, this.checked);">'+ this.fullName**：设置复选按钮，显示在设置的 cookie 名前。并调用设置的 checked()函数，利用 onclick 事件处理器，处理用户的点击事件：如果用户点击了该属性值，即用户选取了该项目，那么用户在显示的时候，就会显示该项目；
- **document.write(result)**：调用 document 对象的 write()方法，开启文档显示用

户的设置。

7. Writeasweblink()函 数

该函数设置 cookie 对象的链接。

在该脚本中，调用设置的函数，判断用户是否选择了系统设置的 cookie，或是用户点击了显示系统设置的 cookie，因而开启新文档显示用户选择的内容。该脚本如下：

```
function WriteAsWebLink()
{
  var result = '';
  if (this.urlpath == "")
  {
    NextHeading = this.fullName;
  }
  else
  {
    if (this.Enabled() || (GetCookie("ViewAll") == "T"))
    {
      if (NextHeading != "")
      {
        result = '<p><dt><strong>' + NextHeading+ '</strong>';
        NextHeading = "";
      }
      result = result + '<dd><a href="' + this.urlpath + '">'+ this.fullName + '</a>';
    }
  }
  document.write(result);
}
```

其中：

- **var result = ''**：设置一个字符串变量;
- **if (this.urlpath == "")**：判断用户是否设置了对象的链接路径;
- **NextHeading = this.fullName; nextheading**：是设置的全局变量，利用该变量来传递分类的标题值。以上的语句意思为如果用户没有设置 path 路径值，就认为用户设置的是分类的表头;
- **Else**：如果用户设置了 cookie 的 path 属性值;
- **if (this.Enabled() || (GetCookie("ViewAll") == "T"))**：调用设置的函数先判断该对象是否处于选择状态或点击了显示按钮，如果是，则执行后面的脚本;
- **if (NextHeading != "")**：如果设置全局变量不为空，则认为分类的表头存在;
- **result = '<p><dt>' + NextHeading+ ''**：加粗显示分类的表头;
- **NextHeading = ""**：清空设置的表头;
- **result = result + '<dd>'+ this.fullName + ''**：设置选择对象的链接;
- **document.write(result)**：开启新文档显示用户的链接。

8. Favelist 数 组

该数组形成一个数据表，管理用户输入的 cookie 数据。其格式如下：

```
var FaveList = new Array();
// Comics Section --------------------
FaveList[1] = new favorite("Comics", "", "");
FaveList[2] = new favorite("Dilbert", "cdilb","http://www.unitedmedia.com/comics/dilbert/");
FaveList[3] = new favorite("Doonesbury", "cdoon","http://www.uexpress.com/cgi-
bin/ups/mainindex.cgi?code=db");
FaveList[4] = new favorite("Mr. Boffo", "cboff","http://www.uexpress.com/cgi-
bin/ups/new_mainindex.cgi?code=mb");
// General News Section -------------
FaveList[5] = new favorite("General News", "", "");
FaveList[6] = new favorite("CNN", "ncnn", "http://www.cnn.com/");
FaveList[7] = new favorite("NPR", "nnpr","http://www.npr.org/news/");
FaveList[8] = new favorite("Boston Globe", "nbos","http://www.boston.com/");
// Computer Industry Section --------
FaveList[9] = new favorite("Computer Industry", "", "");
FaveList[10] = new favorite("PC Week", "ipcw","http://www.pcweek.com/");
FaveList[11] = new favorite("TechWeb", "icmp","http://www.techweb.com/wire/wire.html");
FaveList[12] = new favorite("Netscape", "ntsc","http://devedge.netscape.com/");
FaveList[13] = new favorite("Microsoft", "micr","http://msdn.microsoft.com/");
// Search Engines Section -----------
FaveList[14] = new favorite("Search Engines", "", "");
FaveList[15] = new favorite("Yahoo!", "syah","http://www.yahoo.com/");
FaveList[16] = new favorite("Alta Vista", "sav","http://www.altavista.com/");
FaveList[17] = new favorite("Excite", "sexc","http://www.excite.com/");
// Auction Section ------------------
FaveList[18] = new favorite("Auctions", "", "");
FaveList[19] = new favorite("ebay", "ebay","http://www.ebay.com/");
FaveList[20] = new favorite("Yahoo Auctions", "yhac","http://auctions.yahoo.com/");
// Misc. Section --------------------
FaveList[21] = new favorite("Misc.", "", "");
FaveList[22] = new favorite("Today in History",
"mtih","http://www.thehistorynet.com/today/today.htm");
FaveList[23] = new favorite("Merriam-Webster's Word of the Day","mwod", "http://www.m-
w.com/cgi-bin/mwwod.pl");
FaveList[24] = new favorite("Quotes of the Day",
"mquot","http://www.starlingtech.com/quotes/qotd.html");
```

其中：

- **var FaveList = new Array()**：设置数组对象；
- **FaveList[1] = new favorite("Comics", "", "")**：调用设置的函数来处理用户的设置，将该设置转化为新的对象值，并将该值赋给设置数组，作为该数组的一个元素。

9. Sendoptionspage()函 数

该函数形成一个表单，设置表单按钮，显示选择的信息。

该函数形成的表单由 document 对象的 write()方法构造，它开启新窗口来动态显示。在显示的时候，调用设置的 writeasweblink()函数，检查设置项目的选择状态。然后调用设置的 clearcookie（）函数清除系统设置的项目，显示用户选择的项目。该脚本如下：

```
function SendOptionsPage()
{
  document.write('<h1>显示我的链接</h1>');
  document.write('<form method=post>');

  document.write('<input type=button value="选择" '+ 'onClick="'+'ReloadPage()'+';">');

  for (var i = 1; i < FaveList.length; i++)
    FaveList[i].WriteAsCheckBox();

  document.write('</dl><p>');
  ClearCookie("ViewAll");
  document.write('</form>');
}
```

其中：

- **document.write('<h1>显示我的链接</h1>')**：调用 document 对象的 write()方法显示新文档的提示信息;
- **document.write('<form method=post>')**：设置表单。调用 document 对象的 write()方法在新页面中设置一个表单，该表单以 post 的方法传送;
- **document.write('<input type=button value="选择" '+ 'onClick="'+'Reload-Page()'+';">')**：设置一个按钮，并设置 onclick 事件处理器调用设置的 reload（）函数，显示用户的设置;
- **for (var i = 1; i < FaveList.length; i++)**：设置循环;
- **FaveList[i].WriteAsCheckBox()**：调用设置的函数，检索用户选择的项目，然后在其前面的复选框中划勾。在用户按下显示按钮的时候，显示用户的选择;
- **ClearCookie("ViewAll")**：调用设置的函数，清除系统设置的项目，只显示用户选择的项目;
- **document.write('</form>')**：结束新页面表单的设置。

10. Clearcookie()函 数

该函数按要求清除系统设置的 cookie 项目。

在该脚本中，设置了 cookie 过期的时间为 3 天，如果用户没有选择该项目，系统按默认的项目执行。该脚本如下：

```
function ClearCookie(name)
{
```

```
  var ThreeDays = 3 * 24 * 60 * 60 * 1000;
  var expDate = new Date();
  expDate.setTime (expDate.getTime() - ThreeDays);
  document.cookie = name + "=ImOutOfHere; expires="+ expDate.toGMTString();
}
```

其中：

- **var ThreeDays = 3 * 24 * 60 * 60 * 1000**：设置变量。从其后的计算中可以看出刚好是 3 天时间的毫秒数;
- **var expDate = new Date()**：获取系统当前时间，并将该变量赋给用户设置的变量;
- **expDate.setTime (expDate.getTime() - ThreeDays)**：调用 date 对象的 settime（）函数，设置新的变量。这里设置 cookie 到期的时间;
- **document.cookie = name + "=ImOutOfHere; expires="+ expDate.toGMTString()**：将设置的 cookie 过期时间赋给设置的文档 cookie 变量。

11. Checked()函 数

该函数用来判断用户是否选择了用户设置的项目。如果用户选择了该项目，即该项目前的复选框处于选择状态，则返回一个“checked”变量。否则返回一个空字符。

同时，该函数还是 favorite 对象的方法。

该脚本如下：

```
function Checked ()
{
  if (this.Enabled())
    return "CHECKED ";
  return "";
}
```

其中：

- **if (this.Enabled())**：调用设置的 enabled()方法，探索用户选择的对象是不是用户的检索对象;
- **return "CHECKED "**：如果是，则返回一个“checked”的标志;
- **return ""**：如果不是，则返回一个空字符。

12. loadoptions()函 数

该函数设置当用户按下显示按钮的时候，显示用户选择设置的项目。

该脚本语句在功能的执行中，调用被设置的 Sendoptionspage（）函数。该脚本如下：

```
function LoadOptions()
{
  SetCookieEZ("ShowOptions", "T");
  window.open(document.location.href, "_top", "");
}
```

其中：

- **SetCookieEZ("ShowOptions", "T")**：调用设置的函数 setcookieEZ,设置文档的 cookie 值;
- **window.open(document.location.href, "_top", "")**: 调用 window 对象的 open()方法，显示用户设置的 cookie 项目。

13. Toggleview()函　数

该函数响应用户的请求，并显示用户的设置。

在该脚本中，先调用设置的 getcookie（）函数，检索用户的设置，如果用户的设置存在，再调用设置的 clearcookie 函数来清除用户的设置。然后在设置 cookie 的过期时间为 5 天。再调用 window 对象的 open()方法，将用户的设置显示出来。该脚本如下：

```
function ToggleView()
{
  if (GetCookie("ViewAll") == "T")
  {
    ClearCookie("ViewAll");
  }
  else
  {
    var fiveYears = 5 * 365 * 24 * 60 * 60 * 1000;
    var expDate = new Date();
    expDate.setTime (expDate.getTime() + fiveYears );
    SetCookie("ViewAll", "T", expDate, null, null, false);
  }
  window.open(document.location.href, "_top", "");
}
```

其中：

- **if (GetCookie("ViewAll") == "T")**：调用设置的 getcookie 函数，检索“viewall”函数，如果存在，则返回一个 true 值;
- **ClearCookie("ViewAll")**：调用设置的 clearcookie 函数，取消设置的 cookie 项目。这里响应用户的请求，显示用户的设置情况;
- **Else**：如果用户按下其他的功能按钮;
- **var fiveYears = 5 * 365 * 24 * 60 * 60 * 1000**：设置时间变量。这里设置的是五年时间的毫秒数;
- **var expDate = new Date()**：获得用户系统的当前时间;
- **expDate.setTime (expDate.getTime() + fiveYears)**: 调用 date 对象的 settime()方法，设置新文档的 cookie 的过期日期;
- **SetCookie("ViewAll", "T", expDate, null, null, false)**：设置新文档 cookie 的属性值;
- **window.open(document.location.href, "_top", "")**：调用 window 对象的 open 方法，开启新窗口显示设置的文档。

14. Sendpersonalpage()函 数

该函数显示系统设置的 cookie 来让用户选择。最后显示用户选择的设置。

在该脚本中，动态生成相应的 form 表单，并设置页面按钮，预置相应的事件处理器，并响应用户的请求。

```
function SendPersonalPage()
{
  if (GetCookie("ViewAll") != "T")
    document.write('<h1>你的珍藏:</h1>');
  else
    document.write('<h1>我的链接： </h1>');
    document.write('<form method=post>');
  if (GetCookie("ViewAll") == "T")
  {
    document.write('<input type=button value="显示" '+'onClick="ToggleView();">');
  }
  else
  {
    document.write('<input type=button value="显示我的珍藏 " '+'onClick="ToggleView();">');
  }
  document.write('<input type=button '+ 'value="点击选择显示我的珍藏： '+
'onClick="LoadOptions();">');
  document.write('</form>');
   document.write('<dl>');
  for (var i = 1; i < FaveList.length; i++)
    FaveList[i].WriteAsWebLink();
  document.write('</dl><p>');
}
```

其中：

- **if (GetCookie("ViewAll") != "T")**：检索“viewall”，看该词是否存在系统设置的 cookie 中;
- **document.write('<h1>你的珍藏:</h1>')**：开启新文档，显示标题;
- **else**：如果按钮不为显示系统设置的按钮;
- **document.write('<h1>我的链接： </h1>')**：显示我的链接标题;
- **document.write('<form method=post>')**：设置表单，该表单的传递方式为”post”;
- **if (GetCookie("ViewAll") == "T")**：检索关键词” viewall”;
- **document.write('<input type=button value="显示" '+'onClick="ToggleView();">')**：如果用户页面处于显示用户设置状态，设置按钮，显示用户的设置。这里设置了一个 onclick 事件处理器，调用设置的 toggleview()函数，响应用户的输入;
- **else**：如果检索的字段不存在;
- **document.write('<input type=button value="显示我的珍藏"'+'onClick="ToggleView();">')**：如果检索的字段不存在，则认为用户处于选择状态，这时提示用户选择信息并显示用户的设置。

- **document.write('<input type=button '+ 'value="点击选择显示我的珍藏：'+ 'onClick="LoadOptions();">')**：设置按钮，预制 onclick 事件处理器，开启新文档显示用户的设置;
- **document.write('</form>')**：结束表单的设置;
- **for (var i = 1; i < FaveList.length; i++)**：设置循环，显示用户的检索信息;
- **FaveList[i].WriteAsWebLink();调用设置的函数 writeasweblink()**：设置选择项目的链接信息。

15. Isenable()函　数

该函数调用设置的函数 getcookie()，设置 favorites 的存在状态。如果该词存在，则获取要检索的 cookie 名字。最后，将传回一个函数值，供用户调用。该脚本如下：

```
function isEnabled(name)
{
  var result = false;
  var FaveCookie = GetCookie("Favorites");
  if (FaveCookie != null)
  {
    var searchFor = "<" + name + ">";
    var startOfCookie = FaveCookie.indexOf(searchFor);
    if (startOfCookie != -1)
      result = true;
  }
  return result;
}
```

其中：

- **var result = false**：设置一个为假的逻辑变量;
- **var FaveCookie = GetCookie("Favorites")**：调用设置的 getcookie 函数，检索 favorites 字段;
- **if (FaveCookie != null)**：如果该字段存在;
- **var searchFor = "<" + name + ">"**：设置搜索字段;
- **var startOfCookie = FaveCookie.indexOf(searchFor)**：获得搜索字段的起始位置;
- **if (startOfCookie != -1)**：如果要搜索的字段不存在;
- **result = true**：返回一个 true 值。

16. addfavorite()函　数

该脚本设置 favorite 对象的 enable 属性。

该函数调用设置的函数 isenabed()，判断用户设置的检索字段是否存在，如果不存在，重新设置文档的 cookie 属性值。

```
function AddFavorite(name)
{
  if (!isEnabled(name))
```

```
  {
    var fiveYears = 5 * 365 * 24 * 60 * 60 * 1000;
    var expDate = new Date();
    expDate.setTime (expDate.getTime() + fiveYears );
    SetCookie("Favorites", GetCookie("Favorites")+ "<" + name + ">", expDate, null, null, false);
  }
}
```

其中：

- **if (!isEnabled(name))**：调用设置的函数 isenabled()，探索用户传递的值是否存在于用户设置的 cookie 字段中；
- **var fiveYears = 5 * 365 * 24 * 60 * 60 * 1000**：设置时间变量；
- **var expDate = new Date()**：获取系统当前时间；
- **expDate.setTime (expDate.getTime() + fiveYears)**：设置时间；
- **SetCookie("Favorites", GetCookie("Favorites")+ "<" + name + ">", expDate, null, null, false)**: 设置用户选择项目的过期时间。这一句话，调用设置的 setcookie 函数，设置用户检索到的 cookie 项目的属性值。

17. Clearfavorite()函 数

该函数用来处理未检索到的 cookie 值。由于设置了复选框，让用户选择要显示的 coookie 值。对于没有显示的 cookie 值，就利用该函数来处理这样的情况。该脚本如下：

```
function ClearFavorite(name)
{
  if (isEnabled(name))
  {
    var FaveCookie = GetCookie("Favorites");
    var searchFor = "<" + name + ">";
    var startOfCookie = FaveCookie.indexOf(searchFor);
    var NewFaves = FaveCookie.substring(0, startOfCookie)+
FaveCookie.substring(startOfCookie+searchFor.length,FaveCookie.length);
    var fiveYears = 5 * 365 * 24 * 60 * 60 * 1000;
    var expDate = new Date();
    expDate.setTime (expDate.getTime() + fiveYears );
    SetCookie("Favorites", NewFaves, expDate, null, null, false);
  }
}
```

其中：

- **if (isEnabled(name))**：调用设置的函数，判断用户的检索字段是否被选择；
- **var FaveCookie = GetCookie("Favorites")**：调用设置的函数，获得用户检索字段；
- **var searchFor = "<" + name + ">"**：设置检索字段；
- **var startOfCookie = FaveCookie.indexOf(searchFor)**：获得检索字段的位置；
- **var NewFaves = FaveCookie.substring(0, startOfCookie)+ FaveCookie.substring(startOfCookie+searchFor.length,FaveCookie.length)**：截取获得在设

置的 cookie 字段中的检索字段;

- **var fiveYears = 5 * 365 * 24 * 60 * 60 * 1000**: 设置过期日期;
- **var expDate = new Date()**: 获得系统当前日期;
- **expDate.setTime (expDate.getTime() + fiveYears)**: 设置时间;
- **SetCookie("Favorites", NewFaves, expDate, null, null, false)**: 设置 favorites 对象的属性值。

18. SetFavoriteEnabled()函 数

该函数设 favorites 对象的状态。如果用户未设置状态，则调用设置的 addfavorite 函数来处理，设置选定状态下的 cookie 值的情况。该脚本如下：

```
function SetFavoriteEnabled(name, SetOn)
{
  if (SetOn)
    AddFavorite(name);
  else
    ClearFavorite(name);
}
```

其中：

- **if (SetOn)**: 如果函数传递的该值为真;
- **AddFavorite(name)**: 调用设置的函数 addfavorite 来处理;
- **Else**: 如果上面的值不成立;
- **ClearFavorite(name)**: 调用设置的函数 clearavorite 来处理文档的 cookie 名。

19. ReloadPage()函 数

该函数处理用户设置的功能按钮，显示设置的页面。该脚本调用 window 对象 open（）方法，显示新文档。该脚本如下：

```
function ReloadPage()
{
  window.open(document.location.href, "_top", "");
}
```

最后，在页面的主体中，设置了如下的脚本，在该脚本中，先显示两个按钮，当用户按下按钮的时候，程序调用相应的函数来响应用户的操作。该脚本如下：

```
<script language="JavaScript">
<!--
if (GetCookie("ShowOptions") == "T")
{
  ClearCookie("ShowOptions");
  SendOptionsPage();
}
else
{
  SendPersonalPage();
```

```
}
// End Commented Script -->
</script>
```

其中：

- **if (GetCookie("ShowOptions") == "T")**：检索用户设置的显示项目；
- **ClearCookie("ShowOptions")**：处理用户没有选择的项目；
- **SendOptionsPage()**：显示用户设的选项；
- **Else**：如果用户没有进行选择操作；
- **SendPersonalPage()**：显示系统设置的选项。

第21章 时钟日历

主要内容

- 时间对象的组合
- 日历的实现

本章导读

本章主要通过利用 JavaScript 构造了一个个人珍藏系统，类似于一个个人书签。该实例通过 cookie 的设置，介绍了 cookie 的设置、保存、检索等等。通过详尽的分析，使用户能够掌握 cookie 的设置。

21.62 示例特性

本例利用 JavaScript 语言，并结合 HTML 语言来实现一个万年历书的设置。通过本例的学习，掌握常用日期的设置方法，并利用 JavaScript 来控制页面格式的显示。该示例有如下的特性。

- 日期对象的属性、方法;
- 日期对象的设置;
- 时段的设置与实现;
- 三元函数的设置;
- 中文星期的设置;
- 数组的设置、利用;
- 利用 JavaScript 语言生成页面;
- 时钟的实现;
- 日历的实现。

21.63 源代码

实现如上特性的源代码如下。

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>日历时钟</title>
<script language="JavaScript">

function TOfunc()
  {

    TO = window.setTimeout( "TOfunc()", 1000 );

    var today = new Date();

    document.clock.disp.value =
today.toLocaleString();

    }
</script>
</head>
<body onload="TOfunc()">
```

```
<p  align="center"><font  face=" 楷 体 _GB2312"  color="#0000FF"><big><big><big><big> 日 历
</big></big></big></big></font></p>

<form name="clock">
  <div align="center"><center><p><input type="text" name="disp" value size="20"> <br>
  </p>
  </center></div>
</form>
<script LANGUAGE="JavaScript">
<!--

function getTime() {
    // initialize time-related variables with current time settings
    var now = new Date()
    var hour = now.getHours()
    var minute = now.getMinutes()
    now = null
    var ampm = ""

    // validate hour values    and set value of ampm
    if (hour >= 12) {
        hour -= 12
        ampm = "下午"
    } else
        ampm = "上午"
    hour = (hour == 0) ? 12 : hour

    // add zero digit to a one digit minute
    if (minute < 10)
        minute = "0" + minute // do not parse this number!

    // return time string
    return hour + ":" + minute + " " + ampm
}

function leapYear(year) {
    if (year % 4 == 0) // basic rule
        return true // is leap year
    /* else */ // else not needed when statement is "return"
        return false // is not leap year
}

function getDays(month, year) {
    // create array to hold number of days in each month
    var ar = new Array(12)
    ar[0] = 31 // January
    ar[1] = (leapYear(year)) ? 29 : 28 // February
```

```
    ar[2] = 31 // March
    ar[3] = 30 // April
    ar[4] = 31 // May
    ar[5] = 30 // June
    ar[6] = 31 // July
    ar[7] = 31 // August
    ar[8] = 30 // September
    ar[9] = 31 // October
    ar[10] = 30 // November
    ar[11] = 31 // December

    // return number of days in the specified month (parameter)
    return ar[month]
}

function getMonthName(month) {
    // create array to hold name of each month
    var ar = new Array(12)
    ar[0] = "1 月"
    ar[1] = "2 月"
    ar[2] = "3 月"
    ar[3] = "4 月"
    ar[4] = "5 月"
    ar[5] = "6 月"
    ar[6] = "7 月"
    ar[7] = "8 月"
    ar[8] = "9 月"
    ar[9] = "10 月"
    ar[10] = "11 月"
    ar[11] = "12 月"

    // return name of specified month (parameter)
    return ar[month]
}

function setCal() {
    // standard time attributes
    var now = new Date()
    var year = now.getYear()
    var month = now.getMonth()
    var monthName = getMonthName(month)
    var date = now.getDate()
    now = null

    // create instance of first day of month, and extract the day on which it occurs
    var firstDayInstance = new Date(year, month, 1)
    var firstDay = firstDayInstance.getDay()
    firstDayInstance = null
```

```
        // number of days in current month
        var days = getDays(month, year)

        // call function to draw calendar
        drawCal(firstDay + 1, days, date, monthName, 1900 + year)
    }

    function drawCal(firstDay, lastDate, date, monthName, year) {
        // constant table settings
        var headerHeight = 50 // height of the table's header cell
        var border = 2 // 3D height of table's border
        var cellspacing = 4 // width of table's border
        var headerColor = "midnightblue" // color of table's header
        var headerSize = "-1" // size of tables header font
        var colWidth = 50// width of columns in table
        var dayCellHeight = 10 // height of cells containing days of the week
        var dayColor = "darkblue" // color of font representing week days
        var cellHeight = 20// height of cells representing dates in the calendar
        var todayColor = "red" // color specifying today's date in the calendar
        var timeColor = "purple" // color of font representing current time

        // create basic table structure
    var now = new Date()
        var year = now.getYear()

        var text = "" // initialize accumulative variable to empty string
        text += '<CENTER>'
        text += '<TABLE BORDER=1' + ' CELLSPACING=0' + 'style="font-size: 9pt">' // table
settings
        text += '<TH COLSPAN=7 HEIGHT=' + headerHeight + '>' // create table header cell
        text +=         " // set font for table header
        text +=year+'年'+ monthName + '日历'
        text +=         " // close table header's font settings
        text += '</TH>' // close header cell

        // variables to hold constant settings
        var openCol = '<TD WIDTH=' + colWidth + ' HEIGHT=' + dayCellHeight + '>'
        openCol += '<FONT COLOR="' + dayColor + '">'
        var closeCol = '</FONT></TD>'

        // create array of abbreviated day names
        var weekDay = new Array(7)
        weekDay[0] = "星期天"
        weekDay[1] = "星期一"
        weekDay[2] = "星期二"
        weekDay[3] = "星期三"
        weekDay[4] = "星期四"
```

```
    weekDay[5] = "星期五"
    weekDay[6] = "星期六"
       // create first row of table to set column width and specify week day
    text += '<TR ALIGN="center" VALIGN="center"    style="font-size: 9pt">'
    for (var dayNum = 0; dayNum < 7; ++dayNum) {
       text += openCol + weekDay[dayNum] + closeCol
    }
    text += '</TR>'

    // declaration and initialization of two variables to help with tables
    var digit = 1
    var curCell = 1

    for (var row = 1; row <= Math.ceil((lastDate + firstDay - 1) / 7); ++row) {
       text += '<TR ALIGN="right" VALIGN="top" style="font-size: 9pt">'
       for (var col = 1; col <= 7; ++col) {
              if (digit > lastDate)
                     break
              if (curCell < firstDay) {
                     text += '<TD></TD>';
                     curCell++
              } else {
                     if (digit == date) { // current cell represent today's date
                            text += '<TD HEIGHT=1>'
                            text += '<FONT COLOR="' + todayColor + '">'
                            text += digit
                            text += '</FONT><BR>'
                            text += '<FONT COLOR="' + timeColor + '" SIZE=2 style="font-size: 9pt">'
                            text += '<CENTER>' + getTime() + '</CENTER>'
                            text += '</FONT>'
                            text += '</TD>'
                     } else
                            text += '<TD HEIGHT=' + cellHeight + '>' + digit + '</TD>'
                     digit++
              }
       }
       text += '</TR>'
    }

    // close all basic table tags
    text += '</TABLE>'
    text += '</CENTER>'

    // print accumulative HTML string
    document.write(text)
}

// -->
```

```
</script>

</body>
</html>
```

21.64 功能概述

在浏览器中，该页面的表现情况如图 21-1 所示。

图21-114

可以看出，该页面显示当前的日期、星期，并显示用户打开该页面的时间以及模拟显示时钟。这样，我们就简单地实现了日历与时钟。

21.65 程序详解

在该程序中，大部分语句都是由 JavaScript 语句生成的。调用 JavaScript 语句可以轻易实现动态页面的实现，同时也就可以实现动态日期的显示。

21.65.2 基本脚本结构

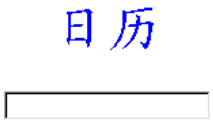


图21-115

该脚本结构就是页面标题的设置。在 IE 中预览效果如图 21-2 所示。

下面是由 HTML 语言实现的基本脚本结构。

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>日历时钟</title>
</head>

<body >

<p  align="center"><font  face=" 楷 体 _GB2312"  color="#0000FF"><big><big><big><big> 日 历
</big></big></big></big></font></p>

<form name="clock">
  <div align="center"><center><p><input type="text" name="shizhong" value size="20"> <br>
  </p>
  </center></div>
</form>
</body>
</html>
```

其中：

- **<body style="font-size: 9pt">**：设置字体样式。这里设置字体的大小为 9pt;
- **<form name="clock">**：设置一个名为“clock”的表单;
- **<input type="text" name="shizhong" value size="20">**：设置一个文本框，长度为 20 个单位。这里我们用来显示设置的时钟。

21.65.3 脚 本 详 解

在这段程序的设计中，我们设置了两段脚本，分别设置时钟的显示和日历表的实现。

1. 时 钟 脚 本

在这段脚本中，我们主要利用一个 settimeout（）的函数来设置时钟的动态显示效果。在每隔一秒钟，系统执行一次设置的函数 tofunc（），这样，在显示时，就可以看到每秒钟动态显示的效果。

在实现该功能特性的时候，我们新设置一个事件对象，以便取得用户系统当前时间，这个时间就是用户开始查看该页面显示的最初时间，然后，将该时间转换为系统当前熟悉

的时间格式。对于国内的用户，则为 xxxx 年 xx 月 xx 日 xx 时 xx 分 xx 秒的格式。

实现这样功能的源代码如下：

```
<script language=”JavaScript”>
function TOfunc()
{
   TO = window.setTimeout( "TOfunc()", 1000
);
   var today = new Date();

   document.clock.disp.value =
today.toLocaleString();

   }

</script>
```

其中：

- **TO = window.setTimeout("TOfunc()", 1000)**：调用 window 对象的 settimeout（）设置每一千毫秒执行一次设置的 tofunc()函数;
- **var today = new Date()**：设置一个 date()对象，并将设置的对象赋给设置的时间参数 today。这里取得系统当前的时间;
- **document.clock.disp.value = today.toLocaleString();toLocaleString()**：为 date 对象的方法，这里将用户系统当前的时间转换为本地习惯的日期字符。然后将转换取得的值赋给设置的文本框，作为文本框 value 属性的值。

2. Onload（）事件处理器

该事件用来处理当页面载入浏览器时，调用系统设置的函数或预定义的方法。在本例中，设置当页面载入浏览器时执行设置的 Tofunc()函数，并显示当前时间。

在设置 onload 事件处理的时候，一定要将设置的语句放在<body >中。页面一经载入，就执行用户设置的方法和函数。

该脚本事件处理如下：

```
<body onload="TOfunc()">
```

对于 onload 时间处理器的设置，一定要将其置于<body>中。

3. 日 历 脚 本

在该脚本中，利用 JavaScript 实现表格，在实现表格的过程中，要求用户一定要熟悉 HTML 语言。所以，要实现高级编程，就必须懂得一定的 HTML 语言。

在这段程序中，设置了中文星期的显示，获取当前日期的星期数。

利用 JavaScript 动态地获取系统当前的日期，显示当前月份、天数以及星期的状况，并形成一个常用的日历。

该脚本还实现了日历格式的显示，利用一些循环语句可以实现对日期显示格式的控

制。

```
<script LANGUAGE="JavaScript">
<!--
function getTime()
{
    // 设置当前日期。
    var now = new Date()
    var hour = now.getHours()
    var minute = now.getMinutes()
    now = null
    var ampm = ""

    // 设置时段。
    if (hour >= 12)
{
        hour -= 12
        ampm = "下午"
    }
else
        ampm = "上午"
    hour = (hour == 0) ? 12 : hour

    // 设置时间。
    if (minute < 10)
        minute = "0" + minute

        return hour + ":" + minute + " " + ampm
}

function leapYear(year)
 {
    if (year % 4 == 0) // 设置闰年
        return true
     else
        return false
}

function getDays(month, year)
 {
    // 设置月份的最后一天。
    var ar = new Array(12)
    ar[0] = 31 // January
    ar[1] = (leapYear(year)) ? 29 : 28 // February
    ar[2] = 31 // March
    ar[3] = 30 // April
    ar[4] = 31 // May
    ar[5] = 30 // June
    ar[6] = 31 // July
```

```
    ar[7] = 31 // August
    ar[8] = 30 // September
    ar[9] = 31 // October
    ar[10] = 30 // November
    ar[11] = 31 // December

        return ar[month]
}

function getMonthName(month)
{
    // 设置中文月份的显示。
    var ar = new Array(12)
    ar[0] = "1 月"
    ar[1] = "2 月"
    ar[2] = "3 月"
    ar[3] = "4 月"
    ar[4] = "5 月"
    ar[5] = "6 月"
    ar[6] = "7 月"
    ar[7] = "8 月"
    ar[8] = "9 月"
    ar[9] = "10 月"
    ar[10] = "11 月"
    ar[11] = "12 月"

    return ar[month]
}

function setCal()
{
    var now = new Date()
    var year = now.getYear()
    var month = now.getMonth()
    var monthName = getMonthName(month)
    var date = now.getDate()
    now = null

    var firstDayInstance = new Date(year, month, 1)
    var firstDay = firstDayInstance.getDay()
    firstDayInstance = null

    // 获得当前日期。
    var days = getDays(month, year)

        drawCal(firstDay + 1, days, date, monthName, 1900 + year)
}
```

```
function drawCal(firstDay, lastDate, date, monthName, year)
{
   // 设置表格。
   var headerHeight = 50
   var border = 2
   var cellspacing = 4
   var headerColor = "midnightblue"
   var headerSize = "-1"
   var colWidth = 50
   var dayCellHeight = 10
   var dayColor = "darkblue"
   var cellHeight = 20
   var todayColor = "red"
   var timeColor = "purple"

   // 获得当前年份。
var now = new Date()
   var year = now.getYear()

   var text = "" // 文本初始化。
   text += '<CENTER>'
   text += '<TABLE BORDER=1'  + ' CELLSPACING=0'  + 'style="font-size: 9pt">'
   text += '<TH COLSPAN=7 HEIGHT=' + headerHeight + '>'  text +=  " // set font for
table header
   text +=year+'年'+ monthName + '日历'
   text +=  "
   text += '</TH>'

     var openCol = '<TD WIDTH=' + colWidth + ' HEIGHT=' + dayCellHeight + '>'
   openCol += '<FONT COLOR="' + dayColor + '">'
   var closeCol = '</FONT></TD>'

   // 设置中文星期。
   var weekDay = new Array(7)
   weekDay[0] = "星期天"
   weekDay[1] = "星期一"
   weekDay[2] = "星期二"
   weekDay[3] = "星期三"
   weekDay[4] = "星期四"
   weekDay[5] = "星期五"
   weekDay[6] = "星期六"

     text += '<TR ALIGN="center" VALIGN="center"  style="font-size: 9pt">'
   for (var dayNum = 0; dayNum < 7; ++dayNum) {
     text += openCol + weekDay[dayNum] + closeCol
   }
   text += '</TR>'
```

```
    var digit = 1
    var curCell = 1

    for (var row = 1; row <= Math.ceil((lastDate + firstDay - 1) / 7); ++row)
{
        text += '<TR ALIGN="right" VALIGN="top" style="font-size: 9pt">'
        for (var col = 1; col <= 7; ++col)
{
                if (digit > lastDate)
                        break
                if (curCell < firstDay)
{
                        text += '<TD></TD>';
                        curCell++
                }
else
{
                        if (digit == date)
{


                                text += '<TD HEIGHT=1>'
                                text += '<FONT COLOR="' + todayColor + '">'
                                text += digit
                                text += '</FONT><BR>'
                                text += '<FONT COLOR="' + timeColor + '" SIZE=2 style="font-size: 9pt">'
                                text += '<CENTER>' + getTime() + '</CENTER>'
                                text += '</FONT>'
                                text += '</TD>'
                        }
 else
                                text += '<TD HEIGHT=' + cellHeight + '>' + digit + '</TD>'
                        digit++
                }
        }
        text += '</TR>'
    }


    text += '</TABLE>'
    text += '</CENTER>'

        document.write(text)
}
setCal()

// -->
</script>
```

4. Gettime（）函　数

利用该函数获取用户系统当前时间，并按国内用户的习惯将时段分为上午和下午，将时钟设置为十二进制的显示方式。

在程序的设计中，要注意释放设置的变量，以提高系统的速度，并优化系统的内存。

```
function getTime()
{
    var now = new Date()
    var hour = now.getHours()
    var minute = now.getMinutes()
    now = null
    var ampm = ""

    if (hour >= 12)
{
        hour -= 12
        ampm = "下午"
    }
 else
        ampm = "上午"
    hour = (hour == 0) ? 12 : hour

        if (minute < 10)
        minute = "0" + minute

    return hour + ":" + minute + " " + ampm
}
```

其中：

- **var now = new Date()**：获得系统当前的时间。并将该值赋给设置的变量 now。这样就建立了一个名为 now 的日期对象;
- **var hour = now.getHours()**：获得系统当前时间中的小时数，并将该值赋给设置的变量;
- **var minute = now.getMinutes()**：获得系统时间时钟的分数，并将该值赋给设置的变量;
- **now = null**：释放 now 对象;
- **var ampm = ""**：设置一个字符串变量，该变量用来设置时段，即将一天分为上午和下午;
- **if (hour >= 12)**：设置判断条件。如果时间不是用十二进制的，设置为十二进制的;
- **hour -= 12**：如果用户系统当前的时间大于 12，就在当前时间的基础上减去 12。设置为十二进制的显示方式;

- **ampm = "下午"**：如果时间超过了十二点，设置 ampm 变量为“下午”；
- **Else**：相对于第一个判断条件，如果用户系统的时间小于 12;
- **ampm = "上午"**：将 ampm 的值设置为‘上午’；
- **hour = (hour == 0) ? 12 : hour**：典型的三元函数的用法。该句的意思为如果时间为零点就设置时间显示为 12，否则显示经过运算的时间;
- **if (minute < 10)**：如果当前时间的分钟数小于了 10。这里主要是将当前日期显示为国内用户所熟悉的两位格式。这样可以控制文本的显示;
- **minute = "0" + minute**：如果小于 10，就在该数前添 0。形成 0x 的格式;
- **return hour + ":" + minute + " " + ampm**：该语句传回系统的小时数、分钟数、以及时段，供以后设置的函数调用和显示。

5. LeapYear（）函　数

该函数用来判断用户当前的年份是否为闰年。如果为闰年就返回一个 true 值。否则为 false。

这里对闰年的判断是相当简单的。用户在完善该程序后可以设置得更完美些。

```
function leapYear(year)
{
   if (year % 4 == 0)
      return true
 else
      return false
}
```

其中：

- **function leapYear(year)**：设置一个函数，判断当前年份，这里将系统当前日期的年份作为参数传递给该函数;
- **if (year % 4 == 0)**：将当前的年份同 4 比较，如果能够被 4 整除，就认为该年为闰年;
- **return true**：如果为闰年，就返回一个真值;
- **Else**：如果该年份不能被 4 整除;
- **return false**：如果不能被 4 整除，则返回一个 false 值，表示该年份不为闰年。

这里做判断是相当重要的，在设置月份的语句中也是相当重要的。因为，闰年的二月份的天数为 29 天，而其他年份的该月天数为 28 天。如果不区分这些，就会使设置的日历产生极大的误差。

6. Getdays（）函　数

利用该函数设置月份的天数。

由于在年份中有闰年之分，一年的月份中有大小之别。所以，设置月份的天数就显得很重要。

在该脚本中设置一个数组，并设置每月的天数。然后将这个值传递出去，以备其他的

函数调用。

在该函数中，传递的参数为系统当前的年份和月份，用来获取系统当月的天数。

```
function getDays(month, year)
 {
   //获得当前月份的天数。
   var ar = new Array(12)

   ar[0] = 31 // January
   ar[1] = (leapYear(year)) ? 29 : 28 // February
   ar[2] = 31 // March
   ar[3] = 30 // April
   ar[4] = 31 // May
   ar[5] = 30 // June
   ar[6] = 31 // July
   ar[7] = 31 // August
   ar[8] = 30 // September
   ar[9] = 31 // October
   ar[10] = 30 // November
   ar[11] = 31 // December

   // 传递参数。
   return ar[month]
}
```

其中：

- **function getDays(month, year)**：设置函数，并传递参数；
- **var ar = new Array(12)**：设置数组，这里定义数组设置月份的天数。最后将设置的数组对象赋给一个变量 ar；
- **ar[0] = 31 // January**：给数组的第一个数赋值 31。设置第一个月的天数为 31 天；
- **ar[1] = (leapYear(year)) ? 29 : 28 // February**：设置第二个月的天数。这里是一个典型的三元函数。在设置过程中，调用设置的判断闰年函数，判断系统传递的年份是否为闰年，如果设置的年份为闰年，则将数字 29 赋给该函数，表示二月的天数为 29 天，否则为 28 天；
- **ar[11] = 31 // December**：设置第十二月的天数为 31 天；
- **return ar[month]**：传递参数。这里将月份的天数传递，以供其他的函数调用；在显示当前月份的天数中，就要利用到该函数。

7. GetMonthName（）函　数

该函数用来设置系统月份的中文显示格式。

在设置的过程中，同样设置数组，将月份的中文名称赋给数组的每一个元素。然后将该值传回，以被其他的函数调用。

在设置的过程中，为了便于被其他的函数调用，这里设置一个参数值“month”以便传递。

```
function getMonthName(month) {
```

```
    // 设置月份数组
    var ar = new Array(12)
    ar[0] = "1 月"
    ar[1] = "2 月"
    ar[2] = "3 月"
    ar[3] = "4 月"
    ar[4] = "5 月"
    ar[5] = "6 月"
    ar[6] = "7 月"
    ar[7] = "8 月"
    ar[8] = "9 月"
    ar[9] = "10 月"
    ar[10] = "11 月"
    ar[11] = "12 月"

    return ar[month]
}
```

其中：

- **getMonthName(month)**：这里将系统的当前月份作为参数传递给函数，利用设置的脚本来处理获得用户系统当前月份的中文名称;
- **var ar = new Array(12)**：设置数组，设置月份的中文名称。考虑到国内用户的习惯，将月份都设置为中文名称;
- **ar[0] = "1 月"**：将第一数组元素设置为一月;
- **ar[11] = "12 月"**：设置第十二个数组的值为 12 月;
- **return ar[month]**：传回参数的值。这样，就可以利用该函数来处理传递来的月份值，然后转换为相应的中文名称，并将该值传回以便被其他程序调用。

8. SetCal（）函　数

该函数设置系统月份的第一天的数值，并将该值联合月份的天数、月份、月份的名称、年份传递给设置的另一个函数。

在设置中，设置一个新的日期对象，然后获得新日期的月份第一天的星期数。这样便于控制、显示。

```
function setCal()
{
    var now = new Date()
    var year = now.getYear()
    var month = now.getMonth()
    var monthName = getMonthName(month)
    var date = now.getDate()
    now = null

    var firstDayInstance = new Date(year, month, 1)
    var firstDay = firstDayInstance.getDay()
```

```
    firstDayInstance = null

    // 获得当前日期。
    var days = getDays(month, year)

      drawCal(firstDay + 1, days, date, monthName, 1900 + year)
}
```

其中：

- **var now = new Date()**：设置新的日期对象。这里获得系统的当前日期;
- **var year = now.getYear()**：获得系统当前年份;
- **var month = now.getMonth()**：获得系统当前月份;
- **var monthName = getMonthName(month)**：调用设置的函数，处理系统当前月份，获得系统当前月份的中文名称;
- **var date = now.getDate()**：获得系统当前的日期;
- **now = nul**：释放设置的变量;
- **var firstDayInstance = new Date(year, month, 1)**：设置一个新的日期对象。在该对象中，将系统的年份、月份和天数 1 作为设置参数;
- **var firstDay = firstDayInstance.getDay()**：获得新日期对象的星期数。该设置相当重要，因为根据常识，我们在设置日历过程中，必须将第一天置于相对应的星期之下;
- **firstDayInstance = null**：释放设置的新日期变量;
- **var days = getDays(month, year)**：调用设置的函数 getdays（）获得系统月份的天数;
- **drawCal(firstDay + 1, days, date, monthName, 1900 + year)**：将参数值传递给设置的函数 drawcal()，处理用户当前的日期。这里传递的参数依次为：系统当前位于的星期数、该月的天数、日期、月分的中文名称、系统当前的年份。

9. Drawcal（）函　数

该函数显示系统设置的日历。

在显示的时候，系统设置了一个表格，以使显示的内容条理清楚。在实现的过程中，系统设置了很多变量，分别来控制表格的显示外观。

考虑到国内用户的习惯，在设置星期的过程中，将外文的星期转换为中文的星期格式。用户可以直接将这段程序拷贝到用户的页面中，以使用户的页面看起来更加专业化。

为了控制日期的显示，系统设置了几个循环语句，在分析时，要注意使用的范围。

该函数的脚本如下：

```
function drawCal(firstDay, lastDate, date, monthName, year)
{
  // 设置表格。
  var headerHeight = 50
  var border = 2
  var cellspacing = 4
```

```
        var headerColor = "midnightblue"
        var headerSize = "-1"
        var colWidth = 50
        var dayCellHeight = 10
        var dayColor = "darkblue"
        var cellHeight = 20
        var todayColor = "red"
        var timeColor = "purple"

        // 获得当前年份。
      var now = new Date()
        var year = now.getYear()

        var text = "" // 文本初始化。
        text += '<CENTER>'
        text += '<TABLE BORDER=1'    + ' CELLSPACING=0'    + 'style="font-size: 9pt">'
        text += '<TH COLSPAN=7 HEIGHT=' + headerHeight + '>'     text +=              " // set font for
table header
        text +=year+'年'+ monthName + '日历'
        text +=          "
        text += '</TH>'

          var openCol = '<TD WIDTH=' + colWidth + ' HEIGHT=' + dayCellHeight + '>'
        openCol += '<FONT COLOR="' + dayColor + '">'
        var closeCol = '</FONT></TD>'

        // 设置中文星期。
        var weekDay = new Array(7)
        weekDay[0] = "星期天"
        weekDay[1] = "星期一"
        weekDay[2] = "星期二"
        weekDay[3] = "星期三"
        weekDay[4] = "星期四"
        weekDay[5] = "星期五"
        weekDay[6] = "星期六"

          text += '<TR ALIGN="center" VALIGN="center"    style="font-size: 9pt">'
        for (var dayNum = 0; dayNum < 7; ++dayNum) {
          text += openCol + weekDay[dayNum] + closeCol
        }
        text += '</TR>'

        var digit = 1
        var curCell = 1

        for (var row = 1; row <= Math.ceil((lastDate + firstDay - 1) / 7); ++row)
      {
          text += '<TR ALIGN="right" VALIGN="top" style="font-size: 9pt">'
```

```
        for (var col = 1; col <= 7; ++col)
    {
            if (digit > lastDate)
                break
            if (curCell < firstDay)
    {
                text += '<TD></TD>';
                curCell++
            }
    else
    {
                if (digit == date)
    {


                    text += '<TD HEIGHT=1>'
                    text += '<FONT COLOR="' + todayColor + '">'
                    text += digit
                    text += '</FONT><BR>'
                    text += '<FONT COLOR="' + timeColor + '" SIZE=2 style="font-size:
9pt">'
                    text += '<CENTER>' + getTime() + '</CENTER>'
                    text += '</FONT>'
                    text += '</TD>'
                }
     else
                    text += '<TD HEIGHT=' + cellHeight + '>' + digit + '</TD>'
                digit++
            }
        }
        text += '</TR>'
      }


      text += '</TABLE>'
      text += '</CENTER>'

        document.write(text)
    }
```

其中：

- **function drawCal(firstDay, lastDate, date, monthName, year)**：设置一个函数，这里传递的参数分别为：第一天的星期数、月份的天数、日期、月份的中文名称、年份；
- **var headerHeight = 50**：设置表格变量。在该语句中设置文档头的尺寸为 50；
- **var border = 2** ：设置表格边框为 2 个单位；
- **var cellspacing = 4** ：设置表元间的距离为 4 个单位；

- **var headerColor = "midnightblue"**：设置表格文档头的颜色为蓝色;
- **var headerSize = "-1"**：设置表格文档头的字体大小;
- **var colWidth = 50**：设置表格的列宽;
- **var dayCellHeight = 10**：设置表元大小;
- **var dayColor = "darkblue"**：设置当前日期的颜色;
- **var cellHeight = 20**：设置表元的高度;
- **var todayColor = "red"**：设置当前天数的颜色;
- **var timeColor = "purple"**：设置当前日期的颜色;
- **var now = new Date()**：取得用户当前日期;
- **var year = now.getYear()**：取得当前年份值;
- **var text = ""** ：设置一个字符串变量，并设置文本初始化;
- **text += '<CENTER>'**：设置表格居中;
- **text+='<TABLE BORDER=1'+'CELLSPACING=0' + 'style="font-size: 9pt">'**：设置表格的外观、表元间的距离以及表格字段字体的大小;
- **text += '<TH COLSPAN=7 HEIGHT=' + headerHeight + '>'text += ''**：设置表格文档头的大小以及位于该位置文档字体的大小;
- **text +=year+'年'+ monthName + '日历'**：设置文档头的内容。在 ie 中显示为用户系统当前的年份和月份;
- **text +=** ："加入空格;
- **text += '</TH>'**：设置行;
- **var openCol = '<TD WIDTH=' + colWidth + ' HEIGHT=' + dayCellHeight + '>'**：设置表格表元的宽度和高度;
- **openCol += '<FONT COLOR="' + dayColor + '">'**：设置表格文档字体的颜色;
- **var closeCol = '</FONT></TD>'**：结束上面的定义;

然后系统再处理星期的显示，国内用户大都习惯中文星期的显示格式，所以利用下面的语句来实现这样的功能。

其中：

- **var weekDay = new Array(7)**：设置一个数组对象，该数组长度为 7，分别设置一个星期的日期;

以下为该数组的初始化。

weekDay[0] = "星期天"，设置第一个数组元素的值为星期天。
weekDay[1] = "星期一"
weekDay[2] = "星期二"
weekDay[3] = "星期三"
weekDay[4] = "星期四"
weekDay[5] = "星期五"
weekDay[6] = "星期六"

- **text += '<TR ALIGN="center" VALIGN="center" style="font-size: 9pt">'**：设置表格行元素的显示方式;
- **for (var dayNum = 0; dayNum < 7; ++dayNum)**：设置循环控制星期的显示。在该设置中，设置一排显示星期数目，所以设置了七次循环;

- **text += openCol + weekDay[dayNum] + closeCol** ： 设置星期的显示格式;
- **text += '</TR>'**: 结束该行元素的定义;
- **var digit = 1**: 定义变量;
- **var curCell = 1**: 定义变量。

这两个变量的定义用来控制文档的显示。在利用中，设置一个比较，如果系统在循环过程中得到的数据小于设置的变量，就进行其他脚本的处理。设置变量的目的，就是为了便于循环的控制。

- **for (var row = 1; row <= Math.ceil((lastDate + firstDay - 1) / 7); ++row)**: 设置循环，控制表格日期显示的行数。在这个语句中，调用预定义的 math 对象的 ceil()方法，该方法返回一个大于或等于设置参数值的值;
- **text += '<TR ALIGN="right" VALIGN="top" style="font-size: 9pt">'**: 设置日期行的显示格式;
- **for (var col = 1; col <= 7; ++col)**: 设置循环，控制最后行的显示情况;
- **if (digit > lastDate)**: 设置日期的判断条件，如果处理了最后一天的数据则停止循环;
- **Break**: 停止该循环，执行下面的语句;
- **if (curCell < firstDay)**: 如果当前表元没有数据。这里处理没有数据的表格情况，事实上，为了显示的美观，系统专门处理这样的情况;
- **text += '<TD></TD>'**: 设置一个空表元;
- **curCell++**: 循环递增;
- **else**: 如果表元有数据;
- **if (digit == date)**: 如果该行只有一个数据;
- **text += '<TD HEIGHT=1>'**: 设置该表元高度为 1;
- **text += '<FONT COLOR="' + todayColor + '">'**: 设置表格元素的显示外观;
- **text += digit**: 设置表格内容;
- **text += '</FONT>
'**: 结束对字体的设置;
- **text += '<FONT COLOR="' + timeColor + '" SIZE=2 style="font-size: 9pt">'**: 设置表元字体的显示外观;
- **text += '<CENTER>' + getTime() + '</CENTER>'**: 设置当前日期居中显示;
- **text += '</FONT>'**: 结束字体的设置;
- **text += '</TD>'**: 结束表元的设置;
- **Else**: 设置其他表元;
- **text += '<TD HEIGHT=' + cellHeight + '>' + digit + '</TD>'**: 设置表元的高度;
- **digit++**: 数据累加;
- **text += '</TR>'**: 结束表格行的设置;
- **text += '</TABLE>'**: 结束表格的设置;
- **text += '</CENTER>'**: 结束表格居中设置;
- **document.write(text)**: 调用 window 的 write()方法，开启新文档显示设置的表格。

第22章 JavaScript 服务器端编程

主要内容

- 服务器端编程概念
- 服务器端编程基础

本章导读

JavaScript 脚本可以利用 CGI 程序访问联机数据库、Internet 服务，或进行其他类型的服务器处理。要实现这些服务就涉及到服务器端脚本（server-side script）的概念。

服务器端的脚本是指哪些脚本的操作更适用于服务器端，其在页面被送往用户浏览器前执行。利用服务器端的脚本可以轻易地实现与数据库通信，发布产品目录。相对于 netscape 服务器来说，服务器端脚本就是使用 livewrite 开发服务器端的脚本与 web 服务器通信的服务方应用程序。而相对于 Microsoft 的服务方来说，就是如今十分流行的 ASP。

考虑到国内广大的用户使用的是 IE 浏览器，下面，我们列举实例讲解 ASP 方面的基础知识。通过该例的学习，使用户可以学会利用 ASP 通过表单与用户交流，收集用户的信息。

22.66 预备知识

22.66.1 CGI

CGI，通用网关接口，是服务器和超文本之间的一个接口程序，负责处理超文本文件与运行在服务器中的程序间的数据交换。

CGI 的工作过程是：

- 用户的浏览器向 WEB SERVER 发出请求;
- WEB SERVER 把这个请求转给对应的 CGI 程序;
- CGI 程序处理完后，把结果转成网页形式，还给 WEB SERVER;
- WEB SERVER 再把这个网页回传给用户。

CGI 很好地实现动态内容，但它有两个主要缺点：

- 一是对用户的每一个请求，CGI 都要产生一个新的进程。在同一时刻发出的请求越多，服务器产生的进程也就越多;
- 另一方面，CGI 的主要编程语言是 C、VB 和 PERL 等语言，对大多数网页开发人员来说，要掌握和精通这些编程语言需要花很长的时间，所以编制和修改 CGI 程序和人工成本很高。

22.66.2 ASP

动态服务器网页（Active Server Pages）是由 Microsoft 公司 1996 年 11 月推出的 Web 应用程序开发的新技术。ASP 属于 ActiveX 技术中的服务器端技术，目前只能使用在 Windows 平台上。由于 Windows 平台市场占有率的迅速增长和 ASP 技术的先进性，ASP 技术正逐步流行起来。

ASP 页面提供了 CGI 程序和 ISAPIDLL 所能提供的所有交互性和功能，同时也避免了许多在 CGI 程序和 ISAPILL 中的问题。使用 CGI 程序和 ISAPIDLL 的一个主要的问题就是它们在 WEB 服务器上使用之前，都需要首先编写和编译。而使用 ASP 页面，文档可以在被送到客户端浏览器之前，由 WEB 服务器自动解释。所需要做的仅仅是将 ASP 页面发布到 WEB 服务器上，根本不需要编译。

1. ASP 的主要特性

- **编程简单**：ASP 使用 Script（描述性语言），不需要编译就执行。ASP 还附带了一些常用的功能组件，如访问人数的计数器、网上广告的播放器、判断用户对某个特定文件是否有权的检查器等等;
- **管理方便**：Script 都嵌在网页语言 HTML 内，脚本的编写与 HTML 的开发一次性完成，管理起来很方便。但 CGI 程序与 HTML 是分离的，而且 CGI 程序的开发与 HTML 的编写是两个完全不同的过程;
- **支持广泛**：ASP 除支持 VBScript、JavaScript，还能以插件的形式支持第三方的语言。如 PERL、TCL 等;
- ASP 从推出至今，虽只有短短的两年多时间，但由于它具有开发简单、功能强和灵活等优点，现在已被广泛接受，成为开发动态网络站点的主要技术之一。ASP 正慢慢成为动态 Web 应用程序开发环境的主流动态服务器。

2. ASP 的工作过程

用户在客户机浏览器上输入一个 URL 地址请求一个页面；服务器接受用户的请求后，调出相应页面；服务器把刚调出的页面送到客房机浏览器；用户通过浏览器把需求数据送给服务器；服务器对需求数据进行处理，取出用户提交的信息。如果需要从数据库得到信息，那么它与数据库连接并从数据库取出数据。运行 ASP 文件，按照用户请求生成一个 HTML 结果页面，服务器把结果页面发送给客户机浏览器。

3. ASP 所含的对象

内建组件，所谓的内建组件是指组件本身建立于 ASP 中，使用时不需要给予定义，即可执行。最常用的有：Application、Session、Response、Request 以及 Server 五种。

外挂组件，ASP 在存取数据库时，经常使用 ADO（ActiveX Data Object）的技术来和 ASP 结合，达到存取数据库的功能。因此，在网页上不但可以建立数据库的网页内容，还可以在网页上执行 SQL（结构化查询语言，Structured Query Language）的操作，用户可以在网页上对数据库进行查询、删除以及新增等操作。

ADO 里三个主要对象为：Connection、Recordset、Command。

4. Application 对　象

ASP 的第一个对象共有两个事件程序。

- **Onstart**：当第一位用户使用系统时，就会启动这个事件程序里的叙述;
- **OnEnd**：当最后一位用户离开系统时，就会启动这个事件程序里的叙述。

Application 对象的其他方法：

- **Application.Lock**：将 Application 对象所含有的信息锁定，不准其他用户来修改;
- **Application.UnLock**：将 Application 对象锁定的信息开放，准其他用户进行修

改的操作。若有在程序里处于被锁定的信息，可以用此方法设定将其回复，那么在执行时间结束或是执行结束时，会自动恢复为不被锁定的状态。

5. Session 对 象

ASP 的第二个对象，跟 Application 对象恰巧相反，是适用于单一用户信息。

两个事件程序：

- **OnStart**：当用户浏览系统时，服务器将会自动产生一个 Session 对象;
- **OnEnd**：当遇到此一叙述，或是当用户离开系统、网页执行时间过期时，都会结束 Session 对象。

和 Application 一样，这两个事件程序都必须在 Global.asa 的文件里使用。

Session 对象的属性：

- **CodePage**：让系统可以处理不同国家的文字，但是如果网站是使用 Windows NT 默认的语言，则不需要设定此属性，就可以处理不同国家的文字了;
- **LCID**：返回用户的国际标准地区识别码;
- **SessionID**：返回用户的代码，这个代码是惟一的，仅能代表一位使用者;
- **Timeout**：设定执行的时间，默认值为 20 分钟;
- **Session 对象的方法**：Abandon，强迫结束 Session。

6. Response 对 象

它是 ASP 的第三个对象，可以让信息显示于浏览器的窗口，将信息传达给用户。最常用的语法，就是在程序输入处输入：Response.Write&User，这个 Write 的属性，就是将后面所接的文字显示在屏幕上。其中 User 是参数的名称，直接显示于屏幕。中间是利用&符号做为串接。

Response 对象的属性：

- **Buffer**：将信息存储至缓冲器，直到服务器将此 ASP 文件处理完毕后，才会转到用户的画面，必须在 ASP 程序代码的第一行中设定;
- **ContentType**：设定传送信息为 HTTP 类型;
- **Expires**：设定网页的期限，一过期限，用户不可以再次浏览此画面;
- **ExpiresAbsolute**：设定网页期限的日期时间，一过期限内的日期时间，用户便不可以再次浏览此画面;
- **Status**：设定 HTTP 协议;
- Response：对象的方法;
- **AddHeader**：设定新表头的名称;
- **AppendToLog**：将一字符串加到 Web Server Log 文件的结尾，但是此一字符串不可以是逗点;
- BinaryWrite：传送二进制的数据;
- **Clear**：将在缓冲器里的信息送到用户的画面;
- **End**：结束 ASP 文件的执行，并且将缓冲器里的信息送到用户的画面;

- **Flush**：立即将缓冲器里的信息送到用户的画面;
- **Redirect**：链接到某一特定的 URL 地址；Write，最常被使用的方法，将某一特定的字符串传达到用户的画面，可以是任何数据类型，但是不可以超过 1022 个字符。

7. Request 对 象

ASP 的第四个对象，用来取得用户所输入的信息，经常和 Response 对象一块使用。

Request 对象的属性：

- TotalBytes：读出的字节总数;
- **Request**：对象的方法;
- **BinaryRead**：读取所输入的数据，并将这些数据存储在 SafeArray 里。

8. Server 对 象

ASP 的第五个对象，总是与其他的对象搭配使用。比如与 ADO 对象搭配，可以与数据库文件产生链接。

Server 对象的属性：

- **ScriptTimeout**：设定 Script 的执行时间，如果超过此时间，那么就会产生错误，并且停止 Script 的执行。

Server 对象的方法：

- CreateObject：定义 ASP 对象;
- **HTMLEncode**：采用 HTML 的编码法;
- **MapPath**：将路径转换为物理路径，但是它不会分辨是否破例有这样一种路径;
- **URLEncode**：采用 URL 的编码法。

9. Global.asa 文 件

每一个 ASP 的程序只能是一个 Global.asa 文件位于根目录里，它用来存放 通用的 Application 对象与 Session 对象的事件程序。因为 FrontPage 98 并没有提供编辑这类文件的位置，所以可以在记事本里编辑，最后存储为 Global.asa 文件。接着将这个文件存储在 Web 根目录下，注意它想在位置的属性必须是可以执行 Script 的位置。依照惯例，将 ASP 这个文件夹设定为可执行 Script 的属性，那么只要将 Global.asa 文件放到这个文件夹里，就可以了。

10. ADO 所含的对象

ADO 包含了三个主要的对象，Connection、Recordset 以及 Command。通过对这些对象的了解与应用，再搭配 ASP（Active Server Pages）对象（Application Session Response Request Server）就可以很容易地制作出一个能够快速存取数据库的网页了。ADO 原名为 ActiveX Data Object。通过与 ASP 的结合，可以在网页里进行 SQL（Structured Query Lang

uage 结构化查询语言）的指令，用户就可以轻轻松松并且快速地存取、增加或是删除数据库里的数据了。快速、简单和节省磁盘空间等特性，使得 ADO 成为十分热门的技巧。ADO 通过与 ODBC 的链接，可以链接许多种类型的数据库，因为 ODBC 本身就支援许多种类型的数据库，比如许多书里使用的 Access，dBase、FoxPro、SQL Serber、Excel 等。最常见的对象有：Connection、ecordset 以及 Command 三者。ADO 再通过 VBScript 或者 JavaScript 语言的技巧，不但可以控制存取数据库，还可以达到一些特殊技巧的变化。

- **Connection 对象**：是 ADO 的第一个对象，是协助 ASP 链接到数据库的一个对象，通过和 Data Source Name 的结合，就可以对所选定的数据库产生链接的通道。

常用属性：

- **Attributes**：设定对象的特性，默认值为 0;
- **CommandTimeout**：设定执行指令的时间若是超出此设定时间，那么就会产生错误信息，并中断执行;
- **ConnetionString**：设定链接的信息，例如设定文件名称（File Name ）、密码（Password）及用户名称（User Name）等;
- **ConnctionTimeout**：设定链接的时间，若是超出此设定时间，那么就会产生错误并中断链接;
- **DefaultDatabase**：默认链接的数 IsolationLevel，设定 Transaction（在线交易）的隔离作用域;
- **Mode**：设定可否更改目前的数据;
- **Provider**：选择 OLE DB 的提供，默认值为 MSDASQL（Microsoft ODBC Provider for OLE DB）;
- **Version**：读取目前使用 ADO 的版本;
- Connection：对象常用的方法;
- **Begin Trans**：用来管理 Transaction（在线交易）的方法之一，目的是开始新交易的操作;
- **CommitTrans**：用来管理 Transaction（在线交易）的操作之一，目的是结束存储目前交易的操作;
- **Execute**：用来执行所设定的操作;
- **Open**：用来建立与数据来源的链接;
- **RollbackTrans**：用来管理 Transaction（在线交易）的操作之一，目的是取消目前交易的操作;
- **Close**：用来关闭目前被打开的链接对象;
- **Recordset**：对象 ADO 的第二个对象，应用于数据的查询作业。Recordset 对象会在所链接的数据库文件里，形成一个指针，指向数据库文件数据，然后可以利用 Recordset 对象所含有的方法上下移动这个指针，移到想要查询的数据记录上。通过 Recordset 对象的帮助，会先将数据库数据存储到 Recordset 里。这样，这个数据库就可以让两个以上的用户同时存储使用了。也就是说，如果不使用 Recordset 对象，那么在同一时间之内，数据库里的数据就只能允许一位用户来进行存取的操作;
- **Recordset**：对象的常用属性：

- **AbsolutePage**：目前数据记录在页数中的位置，位于第几页；
- **AbsolutePosition**：目前数据记录在项数中的位置，位于第几项；
- **ActiveConnection**：设定 Recordset 所属的 Connection 对象；
- **BOF**：数据记录的开头；
- **Bookmark**：设定书签记号；目的是为了记录目前数据记录的位置，以便函数将来在程序中也能快速地回到这个位置的数据记录；
- **CacheSize**：表示暂存于内存中的数据记录数量；
- **CursorType**：Recordset 的游标模式；
- **EditMode**：设定目前数据记录的编辑状态；
- **EOF**：与 BOF 相反，指数据记录的结尾；
- **Filter**：设定过滤条件；
- **LockType**：目前数据记录的锁定状态；
- **MaxRecords**：查询结果的最大记录数；
- **PageSize**：显示数据记录最多记录数；
- **PageCount**：显示数据记录每页的记录数量；
- **RecordCount**：显示数据记录所有的记录数量；
- **Source**：设定数据来源；
- **Status**：显示目前数据记录的状态；
- **Recordset 的方法**：AddNew，在数据库里增加一项新的数据记录；
- **CancelBatch**：取消一个批次的更新记录；
- **CancelUpdate**：取消 Update 方法对数据记录所进行的更新操作；
- **Clone**：复制另一个 Recordset 的数据记录；
- **Close**：关闭已打开的对象；
- **Delete**：删除目前的数据记录；
- **GetRows**：将多项记录读入数组之中；
- **Move**：移动数据记录的位置；
- **MoveFirst**：将目前数据记录指针移动到第一项数据的位置；
- **MoveLast**：将目前数据记录指针移动到最后一项数据的位置；
- **MoveNext**：将目前数据记录指针移动到下一项数据的位置；
- **MovePrevious**：将目前数据记录指针移动到前一项数据的位置；
- **NextRecordset**：清除目前的 Recordset 对象，并且执行下一个指令；
- **Open**：打开指针；
- **Requery**：重新执行查询，并且更新数据记录的数据；
- **Resyne**：与数据库做同步的操作，从数据库更新 Recordset 的数据记录；
- **Supports**：设定是否提供其他特别的功能；
- **Update**：将变动的数据更新到数据库里；
- **UpdateBatch**：将变动的批处理数据更新到数据库里；
- **Command**：对象 ADO 的第三个对象，是用来对数据库传递 SQL 的指令，并对数据库进行存取的操作。可以说 Command 是介于 connection 对象与 Recordset 对象之间，目的就是强化对数据库的操作能力、提高存取数据库的能力；

- **Command**：对象常用的属性；
- **ActiveConnection**：设定与 connection 对象的链接；
- **CommandType**：设定所下达命令的类型；
- CommandText：设定 SQL 指令。

22.66.3 编写服务器端脚本

在 asp 中，使用<% %>符号将 ASP 脚本程序代码包起来，.asp 代码运行于服务器中，执行后产生浏览器可执行的 HTML 代码，然后传给页面浏览者。

下面我们设计一个简单的程序，该程序显示四个不同字体大小的字符“hello world”。其代码如图 22-1 所示。

图22-116

将该页面存为 hello.asp，这里一定要存为*.asp 文件的格式。在浏览器中浏览时效果如图 22-2 所示。

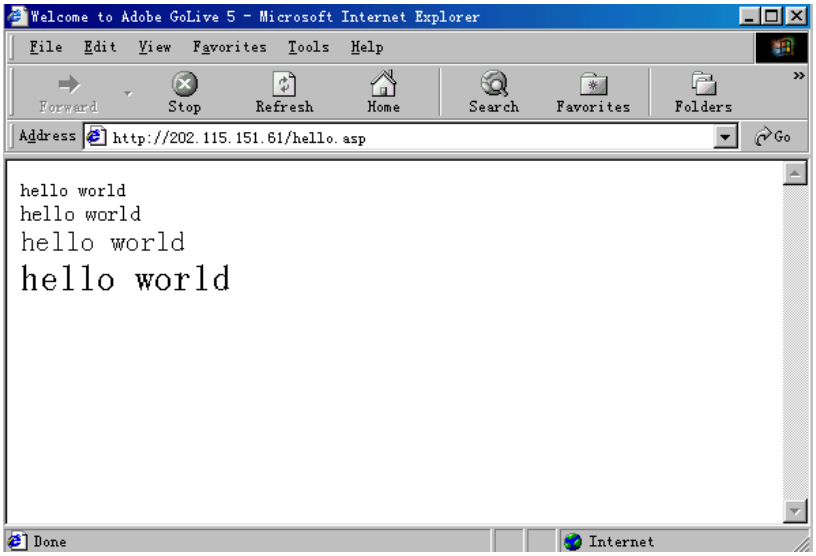


图22-117

22.67 实 例 学 习

在学习了 ASP 的基础知识后，下面举一个简单的实例，实现服务端页面的交互，收集用户的信息，并将用户填写的数据传递给用户。

程序的源代码下：

query.asp 脚本源代码

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
   <title>网络调查</title>
</head>

<body>

<%
response.write"<br><big><strong><blink>请检查填写的数据。</blink></strong></big>"
if request.form("name")<>"" then
response.write"<br>你的姓名是："&request.form("name")
else
response.write"<br>你还没有填写姓名呢，请你填写姓名。"
end if
if request.form("address")<>"" then
response.write"<br>你的地址为："&request.form("address")
else
response.write"<br>你还没有填写地址呢，请填写。"
```

```
end if
response.write"<br>你的性别为："
if request.form("sex")="man" then
response.write"男"
else
response.write"女"
end if
age=""
select case request.form("age")
case 0: age="18 岁以下"
case 1: age="18-25 岁"
case 2: age="26-35 岁"
case 3: age="36-45 岁"
case 4: age="46-65 岁"
case 5: age="66 岁以上"
end select
response.write"<br>你的年龄为："&age
for i=1 to request.form("city").count
if request.form("city")<>"" then
response.write"<br>你要去的城市为："
if request.form("city")(i)="beijing" then
response.write"北京"
end if
else if request.form("city")(i)="shanghai" then
response.write"上海"
else
response.write"其他城市"
end if
end if
next
if request.form("brief")<>"" then
response.write"<br>你的简短留言："&request.form("brief")
end if
 %>
</body>
</html>
```

Query.htm 源代码

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>国庆旅游网上调查</title>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
<script language="JavaScript"><!--
function isrequired(thefield)
{
if( thefield.value=="")
```

```
{
alert("请输入数据！！！");
return false;
}
else
{
return true;
}
}
function queren()
{
if(query.name.value==""||query.address.value=="")
{
query.action="";
alert("请填写好数据！");

return false;
}
else
{
alert("谢谢你的支持！");
}
}
// --></script>
</head>

<body>
<div align="center"><center>

<table border="0" width="748" height="26">
  <tr>
    <td width="188" height="26"><img src="asp.ht1.jpg" alt="wpe3.jpg (1294 bytes)" width="87"
    height="38"></td>
    <td width="548" height="26"><font face="华文中宋" color="#000040"><big><big><big><big>
国庆旅游网上大调查</big></big></big></big></font><hr>
    </td>
  </tr>
</table>
</center></div><div align="center"><center>

<table border="0" width="743" height="240">
  <tr>
    <td width="170" height="240"></td>
    <td width="561" height="240" valign="top"><form method="POST" name="query"
    onsubmit="queren()" action="query.asp">
      <table border="1" width="62%" bgcolor="#C0C0C0">
        <tr>
          <td width="33%">你的姓名：</td>
```

```
                <td width="67%"><input type="text" name="name" onblur="isrequired(query.name) "
size="20"
                size="20"></td>
              </tr>
              <tr>
                <td width="33%">你的地址：</td>
                <td width="67%"><input type="text" name="address" onblur="isrequired(query.address) "
                size="20"></td>
              </tr>
              <tr>
                <td width="33%">你的性别：</td>
                <td width="67%"><input type="radio" value="man" checked name="sex"> 男 <input
type="radio"
                name="sex" value="woman">女</td>
              </tr>
              <tr>
                <td width="33%">你的年龄：</td>
                <td width="67%"><select name="age" size="1">
                  <option value=" 0 ">18 岁以下</option>
                  <option value="1">18-25 岁</option>
                  <option value="2">26-35 岁</option>
                  <option value="3">25-46 岁</option>
                  <option value="4">46-65 岁</option>
                  <option value="5">65 岁以上</option>
                </select></td>
              </tr>
              <tr>
                <td width="33%">你想去的城市</td>
                <td width="67%"><input type="checkbox" name="city" value="beijing" checked>北京
<input
                type="checkbox" name="city" value="shanghai"> 上 海 <input type="checkbox"
name="city"
                value="others">其他城市</td>
              </tr>
              <tr>
                <td width="33%">你的留言：</td>
                <td width="67%"><textarea rows="2" name="brief" cols="20">我想去.....</textarea><p>
</td>
              </tr>
              <tr>
                <td width="33%">  </td>
                <td width="67%"><input type="submit" value="提  交" name="B1">  <input
                type="button" value="取  消" name="B2"></td>
              </tr>
            </table>
          </form>
          </td>
        </tr>
```

```
</table>
</center></div>
</body>
</html>
```

22.68 功能概述

在该示例中，只是简单地处理了表单的数据，在 query.htm 页面中，设置了一些表单控件，收集用户的数据，然后利用表单的 action 属性，传递给设置的 query.asp，将收集到的用户数据显示出来。在 IE 中，query.htm 的表现情况如图 22-3 所示。

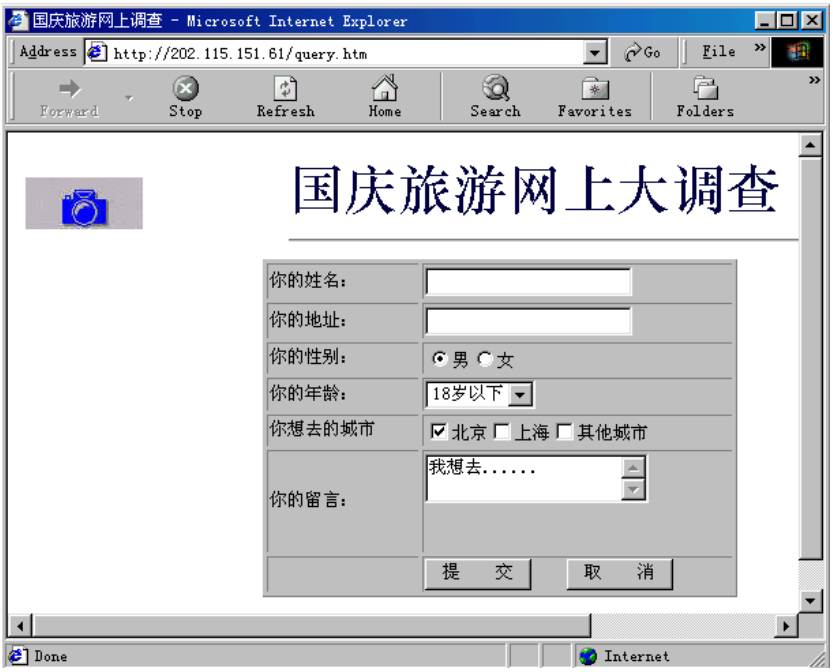


图22-118

在设置页面的时候，要求用户输入用户的姓名和地址，如果用户没有输入，在按下提交按钮的时候，就会提醒用户填写表格。系统就会弹出如图 22-4 所示的提示信息框。

如果用户在文本框中没有输入数据就离开该文本框，系统也会提示用户输入数据，如图 22-5 所示。

图 22-4

图 22-5

如果用户按要求填写了数据并按下提交按钮，系统就会处理用户填写的数据。这里设置的是显示用户填写的数据，如图 22-6、图 22-7 所示。

图 22-6

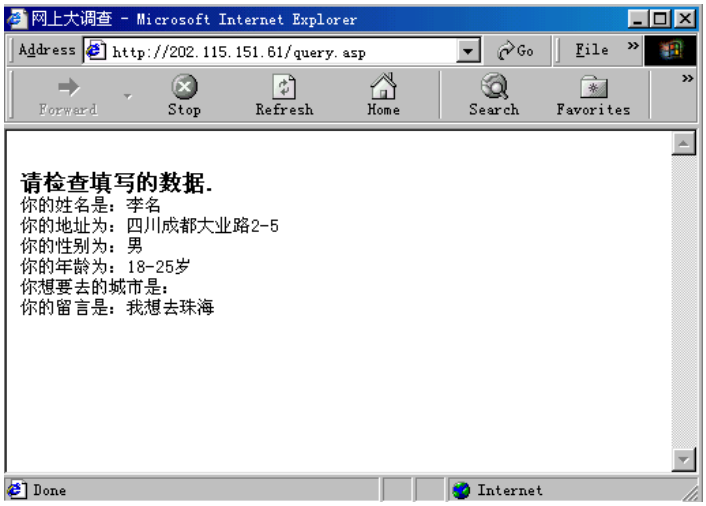


图 22-7

22.69 脚本详解

在该示例的设置中，设置了两个页面，一个页面收集用户的信息，一个页面处理用户输入的数据，在浏览器中显示用户填写的内容。下面详细地分析相应的设置情况。

22.69.1 基本脚本结构

这里设置的是 query.htm 页面的静态结构。在该结构中，设置一个名为 query 的表单，收集用户的信息，包括用户的姓名、地址、性别、年龄、想要去的城市、用户的简短留言等。为了便于脚本的调用，在设置的过程中，给予每个表单控件一个名称。

为了便于格式的控制，设置了表格控制表单控件的位置。其详细的 html 程序结构如下：

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>国庆旅游网上调查</title>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
</head>

<body>
<div align="center"><center>

<table border="0" width="748" height="26">
  <tr>
    <td width="188" height="26"><img src="asp.ht1.jpg" alt="wpe3.jpg (1294 bytes)" width="87"
    height="38"></td>
    <td width="548" height="26"><font face="华文中宋" color="#000040"><big><big><big><big>
国庆旅游网上大调查</big></big></big></big></font><hr>
    </td>
  </tr>
</table>
</center></div><div align="center"><center>

<table border="0" width="743" height="240">
  <tr>
    <td width="170" height="240"></td>
    <td width="561" height="240" valign="top"><form method="POST" name="query"
    action="query.asp">
      <table border="1" width="62%" bgcolor="#C0C0C0">
        <tr>
          <td width="33%">你的姓名：</td>
          <td width="67%"><input type="text" name="name" size="20"></td>
        </tr>
        <tr>
          <td width="33%">你的地址：</td>
          <td width="67%"><input type="text" name="address" size="20"></td>
        </tr>
        <tr>
          <td width="33%">你的性别：</td>
          <td  width="67%"><input  type="radio"  value="man"  checked  name="sex">男<input
type="radio"
          name="sex" value="woman">女</td>
        </tr>
        <tr>
          <td width="33%">你的年龄：</td>
          <td width="67%"><select name="age" size="1">
            <option value=" 0 ">18 岁以下</option>
            <option value="1">18-25 岁</option>
            <option value="2">26-35 岁</option>
```

```
                    <option value="3">25-46 岁</option>
                    <option value="4">46-65 岁</option>
                    <option value="5">65 岁以上</option>
                  </select></td>
                </tr>
                <tr>
                  <td width="33%">你想去的城市</td>
                  <td width="67%"><input type="checkbox" name="city" value="beijing" checked>北京
<input
                  type="checkbox" name="city" value="shanghai"> 上 海 <input type="checkbox"
name="city"
                  value="others">其他城市</td>
                </tr>
                <tr>
                  <td width="33%">你的留言：</td>
                  <td width="67%"><textarea rows="2" name="brief" cols="20"> 我 想
去.....</textarea><p> </td>
                </tr>
                <tr>
                  <td width="33%"> </td>
                  <td width="67%"><input type="submit" value="提 交" name="B1"> <input
                  type="reset" value="取 消" name="B2"></td>
                </tr>
              </table>
            </form>
            </td>
          </tr>
        </table>
        </center>
        </div>
        </body>
        </html>
```

其中：

- <form method="POST" name="query" action="query.asp">：设置表单，名称为 query，设置处理表单的动作为 query.asp;
- **<input type="text" name="name" size="20">**：设置一个名为 name 的文本框，该文本框的大小为 20 个字节。type="text"，指名设置的表单控件为文本框;
- **<input type="radio" value="man" checked name="sex">**：设置一个名为 sex 的单选按钮组，在该单选按钮组中，设置的值为 “man”（ value="man"），并设置该单选按钮为预选状态（checked）。type="radio"指明该表单控件为单选按钮;
- <input type="checkbox" name="city" value="shanghai">：设置一个名为 city 的复选框按钮组。在该语句中，设置该复选框的值为 shanghai;
- （value="shanghai"），type="checkbox"：指明该设置为复选框。

在以下的语句中，设置一个下拉菜单让用户选择用户的年龄段。

```
<select name="age" size="1">
```

```
    <option value=" 0 ">18 岁以下</option>
    <option value="1">18-25 岁</option>
    <option value="2">26-35 岁</option>
    <option value="3">25-46 岁</option>
    <option value="4">46-65 岁</option>
    <option value="5">65 岁以上</option>
</select>
```

其中：

- **<select name="age" size="1">**：该语句指明设置的表单控件为下拉菜单，名称为“age”；
- **<option value="** 0 **">18 岁以下</option>**：设置下拉菜单的项目。这里指定值为 0 的情况为显示的 18 岁以下的情况。指定的值（value=" 0 "）在利用脚本调用的过程中是容易控制和调用的;
- **</select>**：结束下拉菜单的定义;
- **<textarea rows="2" name="brief" cols="20">我想去.....</textarea>**：该语句设置一个滚动文本框，让用户输入简短的留言。设置该滚动文本框的名称为“brief”。在显示的时候，只显示两行（rows="2"），20 个字符（cols="20"）；
- <input type="submit" value="提　　交" name="B1">：该语句设置一个提交按钮。

22.69.2 客户端脚本

在该页面中，设置了客户端脚本，处理用户的输入，以减小服务器端用户数据的处理负担。

在该脚本中，主要设置了让用户填写数据并验证数据。如果用户没有完整地填写，就提示用户填写数据，否则，系统不会处理用户的输入。

该脚本数据如下:

```
<script language="JavaScript">
<!--
function isrequired(thefield)
{
if( thefield.value=="")
{
alert("请输入数据！！！");
return false;
}
else
{
return true;
}
}


function queren()
```

```
{
if(query.name.value==""||query.address.value=="")
{
query.action="";
alert("请填写好数据！ ");

return false;
}
else
{
alert("谢谢你的支持！ ");
}
}

// -->
</script>
```

其中各函数介绍如下。

1. Isrequired（）函　数

该函数用来设置系统必填的数据。在页面的设置过程中，为了让用户填写数据以减少服务器的负担，设置了该函数，当用户在该文本框中形成插入点而没有填写数据就要离开该文本框时，就会提醒用户输入数据。

为了便于参数的传递和函数的调用，在设置函数的过程中，设置了一个 arg 参数：thefield.。

该脚本如下：

```
function isrequired(thefield)
{
if( thefield.value=="")
{
alert("请输入数据！！！ ");
return false;
}
else
{
return true;
}
}
```

其中：

- **if(thefield.value=="")**：判断用户是否输入数据。其中，thefield 取得表单对象，thefield.value，设置表单对象的值。该语句确认用户有没有输入数据;
- **alert("请输入数据！！！ ")**：如果用户没有输入数据，就提示用户输入数据;
- **return false**：传递一个 false 值，以便函数的调用;
- **Else**：如果用户输入了数据。该值与上一个 if 语句相对;
- **return true**：传递一个 true 值。

2. Onblur（）事件处理器

Onblur 事件处理器用来处理当窗口、文档、帧组、表单控件失去输入焦点的时候，调用设置的函数或系统预定义的方法来处理用户的事件。在该脚本中，设置了当用户在姓名文本框或是地址文本框中形成插入点时输入数据，在离开这两个文本框的时候，调用设置的 isrequired()函数，处理用户输入的数据。

```
<input type="text" name="name" onblur="isrequired(query.name) " size="20"
        size="20">
input type="text" name="address" onblur="isrequired(query.address) ">
```

其中：

- **onblur="isrequired(query.name)"**：将 query 表单的名为 name 文本框的数据传递给设置的函数 isrequired()，供该函数处理。

3. Queren（）函　数

该函数用来处理当用户按下提交按钮的时候，检查用户是否填写了设置的表单控件，如果用户没有填写，则不发送该表单，如果填写了，就谢谢用户的支持。

该脚本如下：

```
function queren()
{
if(query.name.value==""||query.address.value=="")
{
query.action="";
alert("请填写好数据！ ");

return false;
}
else
{
alert("谢谢你的支持！ ");
}
}
```

其中：

- if(query.name.value==""||query.address.value==""): 判断用户是否填写了设置的姓名文本框和地址文本框;
- **query.action="";**: 如果用户没有填写，设置 action 为空;
- **alert("请填写好数据！ ")**: 系统弹出一个对话框，提示用户填写数据;
- **return false**: 传递一个 flase 值。以便其他函数调用;
- **Else**: 如果用户填写了表单;
- alert("谢谢你的支持！ "): 提示用户，谢谢用户的支持。

4. Onsubmit（）事件处理器

该事件处理器用来处理当用户提交表单的时候，调用设置的函数或系统预定义的方法处理表单的输入或选择。在该脚本中，设置当用户按下提交按钮的时候，调用设置的函数queren（），检查用户是否填写了数据。

该事件的设置一般位于<form >中。

```
<form method="POST" name="query"
      onsubmit="queren()" action="query.asp">
```

22.69.3 服务端脚本详解

在 query.asp 页面中，设置了处理用户输入的数据。脚本代码如下：

```
<html>
<head>
   <title>网络调查</title>
</head>

<body>

<%
response.write"<br><big><strong><blink>请检查填写的数据。</blink></strong></big>"
if request.form("name")<>"" then
response.write"<br>你的姓名是："&request.form("name")
else
response.write"<br>你还没有填写姓名呢，请你填写姓名。"
end if
if request.form("address")<>"" then
response.write"<br>你的地址为："&request.form("address")
else
response.write"<br>你还没有填写地址呢，请填写。"
end if
response.write"<br>你的性别为："
if request.form("sex")="man" then
response.write"男"
else
response.write"女"
end if
age=""
select case request.form("age")
case 0: age="18 岁以下"
case 1: age="18-25 岁"
case 2: age="26-35 岁"
case 3: age="36-45 岁"
case 4: age="46-65 岁"
case 5: age="66 岁以上"
```

```
end select
response.write"<br>你的年龄为："&age
for i=1 to request.form("city").count
if request.form("city")<>"" then
response.write"<br>你要去的城市为："
if request.form("city")(i)="beijing" then
response.write"北京"
end if
else if request.form("city")(i)="shanghai" then
response.write"上海"
else
response.write"其他城市"
end if
end if
next
if request.form("brief")<>"" then
response.write"<br>你的简短留言："&request.form("brief")
end if
 %>
</body>
</html>
```

其中：

- **<%**：设置 asp 文件的标志。前面已经讲过，asp 文件一般是放在**<%%>**之间;
- response.write"
<big><blink>**请检查填写的数据**。/blink> </big>"，response：设置用户输出数据。该语句设置输入主体的格式，提示用户检查填写的数据;
- **if request.form("name")<>"" then**：典型的 asp 文件应用格式。该语句用来判断用户是否输入了数据。其中，request.form()，当用户设置的表单的传递方法为 post 的时候，利用该语句取得用户输入的数据。其中“name “，设置一个数组坐标，告诉 asp 文档，取得值为用户设置的姓名文本框的值。如果用户输入的数据不为空，则执行如下的语句;
- **response.write"
你的姓名是："&request.form("name")**：调用 response 对象的 write 方法，在新文档中写入数据。这里写入的数据为用户在起始页面设置的姓名文本框的值（request.form("name")）;
- **else**：如果用户没有输入数据;
- response.write"
你还没有填写姓名呢，请你填写姓名。"：在利用 asp 产生的文档中写入提示信息，让用户写入数据;
- **end if**：该语句结束判断。在 asp 页面编程中，语句的设置和检查是相当严格的，if 和 end if 的配套是要一一匹配的。有 if 就必有 end if 来结束;
- **if request.form("address")<>"" then**：该语句的意义同上，只是这里设置的是用户的地址值。在这里提醒用户的是，在设置了 if 语句后，一定要跟一个 then 关键字。否则在调试程序的时候就会报错;

- response.write"
你的地址为："&request.form("address"): 在新文档中写入用户填写的数据:用户的地址;
- **else**: 如果用户没有输入地址这样的数据;
- **response.write"
你还没有填写地址呢，请填写。"**: 如果用户没有输入地址数据，提示用户没有输入地址数据，请用户填写相应的数据;
- **end if**: 结束该语句的判断。如果不跟这样的语句来结束用户的定义，系统就会报错。

以上向我们提供了服务器端数据校验的模式。用户在处理表单的过程中，可以直接按照上面的格式套用。事实上，客户端、服务器端数据的处理的原理都差不多。只是在设置的过程中调用的语句不同罢了。

- response.write"
你的性别为：": 在新文档中写入用户的数据：用户选择的性别;
- **if request.form("sex")="man" then**: 这里做这样的处理，主要是考虑到中文用户，设置的页面也为中文的字体。该语句来获得用户的性别值，如果用户设置的性别值为“man”的话，那么就在文档中输入数据“男”;
- response.write"男": 在新文档中输入“男”;
- **else**: 如果用户设置的性别值为其他数据的话;
- response.write"女": 在新文档中输入“女”;
- **end if**: 结束这一模块的定义。

以上的格式，向我们展示了如何处理外文格式为用户所在环境支持的格式。这里将用户不熟悉的英文格式统一为用户熟悉的中文格式环境。

- **age=""**: 设置一个 age 的字符串变量。定义这样的变量，主要是为了处理用户选择的年龄;
- **select case request.form("age")**: 该语句利用 select case 判断用户选择的年龄。利用该语句来判断设置条件较多的时候，取代 if 语句，以使语句简洁，程序更具可读性;
- **case 0: age="18 岁以下"**: 表示如果用户选择了第一个选项，则系统获得一个 0 值，这时再返回去，显示系统选择的第一项值。将 age 值设置为“18 岁以下”;
- **case 1: age="18-25 岁"**: 如果 request.form("age")返回一个值为 1，则将该值所代表的年龄的值赋给 age。这里年龄的值为“18-25 岁”;
- **case 2: age="26-35 岁"**: 如果 request.form("age")返回一个值为 2，则将该值所代表的年龄的值赋给 age。这里年龄的值为“26-35 岁”;
- **case 3: age="36-45 岁"**: 如果 request.form("age")返回一个值为 3，则将该值所代表的年龄的值赋给 age。这里年龄的值为“36-45 岁”;
- **case 4: age="46-65 岁"**: 如果 request.form("age")返回一个值为 4，则将该值所代表的年龄的值赋给 age。这里年龄的值为“46-65 岁”;
- **case 5: age="66 岁以上"**: 如果 request.form("age")返回一个值为 5，则将该值所代表的年龄的值赋给 age。这里年龄的值为“66 岁以上”;
- **end select**: 结束 select case 的设置;
- response.write"
你的年龄为："&age: 在新文档中写入用户选择的年龄段，

显示用户的年龄数目;

- **for i=1 to request.form("city").count**：设置循环。request.form("city").count 取得用户设置的复选框的个数;
- **if request.form("city")<>"" then**：如果用户选取了设置的复选框，这样就会返回一个值。该语句就是判断用户是否选择了复选框其中之一;
- **response.write"
你要去的城市为："**：如果用户选择了，就显示用户显示的信息。由于页面的设计者设置的是获取用户的去向信息，所以我们知道用户是在选择用户要去的城市，所以，在新文档中写入文本用户要去的城市;

考虑到国内用户的习惯，以下的语句将设置的英文值转换为中文值。

- **if request.form("city")(i)="beijing" then**：如果表单返回的值为“Beijing”，即用户选择的是去北京的话，则写入“北京”两个字;
- response.write"北京"：在新文档中写入北京两个字;
- **end if**：结束一个 if 语句的设置;
- else if request.form("city")(i)="shanghai" then：这一句的意义功能同上。只是注意这里 else if 的利用;
- response.write"上海"：写入“上海”;
- **else**：同上面的 if 语句相对，设置意外的情况;
- **response.write**："其他城市"，写入其他城市的提示;
- **end if**：结束中文文档转换的设置;
- **end if**：结束用户选择要去城市的设置;
- **next**：继续循环，直到循环条件的结束;
- if request.form("brief")<>"" then：该语句获取用户输入的简短留言;
- response.write"
你的简短留言："&request.form("brief")：在新文档中写入用户设置的简短留言;
- **end if**：结束该模块的设置;
- **%>**：asp 语句模块结束的标志。在前面已经讲了，asp 语句以“<%”开始，以“%>”语句结束。

第23章 网络安全性

主要内容

- 安 全 漏 洞
- 安全的预防

本章导读

电脑网络由于其地位的特殊性，一直为大众所关注，特别是网络信息和用户隐私等问题。但是由于其传输和控制的难度，使得 Internet 成为一个巨大的安全弱点。正如现实的生活一样，不幸的事情常有发生：机密数据的丢失、系统遭受攻击、页面被黑客攻击、数据遭病毒破坏等等。总之，传输到网络中的信息都会受到伤害。

23.70 安全性破坏的种类

将计算机联接到网络上，特别是 www 上，就会遇到如下安全的破坏：

- 私人或机密数据被盗;
- 数据被修改;
- 数据破坏;
- 病毒入侵、特别是通过网络传输的病毒;
- 页面被黑客攻击;
- 服务器遭受攻击。

23.71 安 全 服 务

由于遭受到以上不安全因素，所以对于 Web 管理人员来说，应该从如下方面入手，解决系统的安全问题。

对敏感数据进行保护，特别是要对数据备份和加密。对于敏感的商业数据，应该要更加注意保护。

定期查毒和杀毒。给服务器装入一个防病毒的软件包。

服务器的管理。要放弃 guest 帐号、定期地改变系统管理员的帐号和密码、选择合适的密码、取消不必要的协议、不要让用户浏览系统的目录、定期地查看系统的记录。

不要让非管理人员利用你的系统。

除了一般的网络安全事物处理，还要从以下几个方面考虑：

- 在 CGI 方面的安全性

公共网关接口 CGI（COMMON GATEWAY INTERFACE）例程是为把相互作用和智能提供给 WEB 漫游者而运行在服务器上的程序。由于这些程序使用服务器的 CPU 来执行操作，因此有可能让某些人故意把特定的数据当做 CGI 请求送入 CPU，而事实上这些数据流可能是病毒或其他非法送入要求处理的程序。

最容易且最流行的阻止 CGI 攻击的方法是设计处理器来终止所有的 CGI 活动（这些活动至多只维持 3S 左右）。这样一来，合法的用户可以通过处理器来执行 CGI 程序，而那些企图闯入的人将被阻止在外（他们的程序可能需要额外的时间）。此外还可以把 CGI 脚本保存在非公共的目录中，或用 CGI 包装（加密 CGI 数据流的程序）来阻止入侵。

- 配置安全的服务器
- 安全的操作系统

一般来说，操作系统越强大，伸缩性越好，但越容易遭受攻击。

- Web 服务器

如果服务器提供的功能越多，安全漏洞就越大。简单的服务器处理静态数据，但安全性好得多。

- 目录管理

为了最大地加强安全，必须将根目录和服务器根目录严格地区分开来，设置服务器目录的权限。同时在 Web 目录的设置中一般不要设置主目录的绝对路径为 Web 服务器缺省的主目录。