全面理解 JavaScript

[美] Don Gosselin 著

马雷 李宝东 李雄成 翻译

宋瀚涛 审校

THOMSON COURSE TECHNOLOGY 汤姆森学习出版集团

清华大学出版社

(京)新登字 158 号

著作权合同登记号：01-2001-4061

内 容 简 介

本书从 HTML 以及编程的基础知识和方法开始介绍，逐步由浅入深，直到如何使用 JavaScript 进行数据库和嵌入数据的操作等比较高级的技术。不管您是否接触过 HTML 或其他编程语言，都能够在本书的指导下逐步地了解和掌握 JavaScript 语言。

本书充分考虑到了适用读者大部分为初级的编程人员，所以对学习过程中的相关概念都进行了详细的阐述和解释，避免了再去查阅其他相关书籍带来的麻烦。并且，本书从始至终都采用概念阐述和实例练习相结合的方式，在了解概念的基础上，通过实例应用，自己动手实践，加深了对概念的理解，逐步培养起了应用 JavaScript 语言的动手能力。

本书的适用读者范围，除了初学者之外，对于那些从事 Web 编程以及技术服务的工程技术人员、用户，本书也是一本很好的参考读物。

Comprehensive JavaScript
Don Gosselin

ISBN 0-619-01555-1
First published by Course Technology, an imprint of Thomson Learning, United States of America. Reprinted for the People's Republic of China by Thomson Learning Asia and Tsinghua University Press under the authorization of Thomson Learning. No Part of this book may be reproduced in any form without the express written permission of Thomson Learning Asia and Tsinghua University Press.

本书中文简体字版由美国 Thomason Learning 公司授权清华大学出版社出版。未经出版者书面允许不得以任何方式复制或抄袭本书内容。

书　　名：全面理解 JavaScript
作　　者：[美] Don Gosselin　著
翻　　译：马　雷　李宝东　李雄成
审　　校：宋瀚涛
出 版 者：清华大学出版社（北京清华大学学研大厦，邮编 100084）
　　http://www.tup.tsinghua.edu.cn
责任编辑：许存权
印 刷 者：
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　印张：38.75　字数：895 千字
版　　次：2002 年 4 月第 1 版　2002 年 4 月第 1 次印刷
书　　号：ISBN 7-302-05101-1/TP・2983
印　　数：0001～5000
定　　价：74.00 元

译 者 序

Netscape 开发的 JavaScript 语言是一种描述性的脚本语言，可以非常自由地被嵌入到 HTML 文件之中。JavaScript 的成功开发确实是一件令人振奋的事情，它的应用使静态的 HTML 页面呈现出了绚丽多彩的动态特性，同时也赋予了 Web 页面更强大的应用功能。例如它的应用提高了和用户的交互性能，增强了客户端 Web 页面的处理能力，减轻了服务器端的负担，并提供了强大的数据库操作能力。由于传统的 Client/Server 模式逐渐向 Browser/Server 模式转变，企业对于 Web 应用的动态性和功能性的要求越来越高，而 JavaScript 顺应了 Web 发展的这种潮流，它是每个 Web 开发人员都必须了解和掌握的一种 Web 开发语言。正是由于对该语言的这种需求，促成了本书的推出。

对于以前从未接触过 Web 应用的初学者来说，本书是一本很好的入门教材。

本书从 HTML 以及编程的基础知识和方法开始介绍，逐步由浅入深，直到如何使用 JavaScript 进行数据库和嵌入数据的操作等比较高级的技术。不管是否接触过 HTML 或其他编程语言，都能够在本书的指导下逐步地了解和掌握 JavaScript 语言。

本书在编写的过程中，充分考虑到了适用读者大部分为初级的编程人员，所以对学习过程中的相关概念都进行了详细的阐述和解释，这就避免了再去查阅其他相关书籍带来的麻烦。并且，本书从始至终都采用概念阐述和实例练习相结合的方式，在了解概念的基础上，通过实例应用，自己动手实践，加深了对概念的理解，逐步培养起了应用 JavaScript 语言的动手能力。

本书的适用读者范围很广，除了初学者之外，对于那些从事 Web 编程以及技术服务的工程技术人员、用户，本书也是一本很好的基础性参考读物。

本书由北京理工大学马雷、李宝东、李雄成翻译，由北京理工大学宋瀚涛教授审校。由于本书涉及的概念广，内容新，翻译有一定难度，加上译者的水平有限，书中不妥或错误之处，欢迎广大读者批评指正。

译　　者

2001 年 9 月于北京

前　言

本书对需要用 JavaScript 编程语言开发 Web 应用的初级程序员提供指导。本书所讲述的 JavaScript 语言兼容于 Netscape JavaScript 1.2，在 Navigator 4 以上或 Internet Explorer 4 以上的版本中支持该版本的 JavaScript。本书假设读者并没有关于编程语言或 HTML 的知识。

组织和指导

本书开始将向读者介绍基本的 JavaScript 编程概念，同时演示实现的语法。World Wide Web、HTML 和 JavaScript 在第 1 章中介绍，随后介绍编程逻辑方法和调试方法。然后，对变量、函数、对象和事件的介绍将为读者理解更深层的概念和技术提供一个框架，同时让读者尝试更广泛的示例程序。第 3 章和第 4 章将教给读者关于数据类型、运算符、控制结构和语句的基础知识。第 5 章是关于窗口和帧的概念。第 6 章展示如何用 JavaScript 创建和操纵表单。在第 7 章里，读者将通过动画了解关于 DHTML 的知识。

在第 8 至 12 章介绍了更深层次的主题。第 8 章介绍如何用 Cookies 技术保存状态信息，以及 JavaScript 在安全性上的特性，演示了如何使用数字认证。第 9 章详细地讨论了如何调试程序，并介绍了两个调试工具：JavaScript Debugger for Navigator 和 Script Debugger for Internet Explorer。第 10 章和第 11 章介绍服务器端的 JavaScript。第 10 章里介绍了部分关于 LiveWire 应用、活动服务器页面和如何创建客户订单。第 11 章继续介绍服务器端的应用，如何使用 LiveWire 和活动服务器页面操作数据库。第 12 章提供关于 Java 小应用程序和嵌入数据的详细知识，包括用 LiveConnect、插件和 ActiveX 控件等。

本书将文字的阐述和相关概念的逐步练习结合起来，加强了对内容的理解和记忆。HTML 的标签和语法是贯穿全书必需的基本知识。

使用本书的读者可以从基础起建立一个应用，这种方式胜过利用重写的代码。利用这种方式可以对使用 JavaScript 进行 Web 编程有更深的理解。当读完本书，将会熟悉如何创建和修改简单的 JavaScript 应用，以及如何利用工具创建更复杂的应用。不管读者是继续学习 JavaScript，还是转而学习其他的脚本语言或面向对象的语言，比如 C++或 Visual Basic，在本书中所获得的关于编程的基础知识还是很有用的。

本书有别于其他的讲述 JavaScript 语言的书籍，主要的区别在以下方面：

- ✧ 特别为读者设计，并不需要读者有编程或 HTML 的经验和知识。
- ✧ 例子的代码简短，一个例子演示一个概念。
- ✧ 对变量、函数、对象和事件，在本书中提早介绍，以便更好地了解本书后部分的

更深层次的概念和技术，并且开始就接触复杂的工程。

- ✧ 通过逐步的练习来阐述内容，文字阐述和练习交叉进行。
- ✧ 必需的 HTML 标签和语法贯穿全书。
- ✧ 从基础开始创建 JavaScript 应用，读者会对如何建立复杂的程序有一个清晰的了解。
- ✧ 用简单易学的方法教授编程技术。

特点

本书的优秀之处还在于以下的几点：

- ✧ “导读”页：本页和 Course Technology 对用户的承诺精神是一致的，该页有助于教师的教学。硬件、软件和一些默认配置放在同一个地方，这样可以帮助教师节省时间并且减少不必要的烦恼。
- ✧ 案例教学的方法：在每章后面都有可能在应用中遇到的与编程有关的问题。所有的案例后面都有解决的方法，并对其进行详细的说明。在学习如何创建一个应用之前，先演示一个完整的应用，这样会做到有的放矢，对学习相当有益。先演示在读完一章后将要创建应用的类型，读者将更有兴趣去学习。因为他们可以看到自己将要学习的编程概念是如何应用的。同时，也明白了这些概念的重要性。
- ✧ 按部就班的教学方法：本书采用这种独特的教育方法，有助于跟上进度。读者一直需要编写程序代码来解决书中提到的问题。在这个过程中，本书一直指导并让他们知道自己处在解决问题过程中的哪个阶段。大量的示例指导读者创建有用的、可以运行的程序。
- ✧ 提示：本书中的这些提示提供了很多额外的信息，例如：执行一个过程的另一种方法、一种技术的背景信息、需要注意的常见错误、调试的技术，或者提示一个可以找到更多信息的网站地址。
- ✧ 总结：在每章的后面都有一个总结，扼要地重述每个章节里的编程概念和命令。
- ✧ 回顾问题：每章都包括有意义的、概念性的回顾问题，可以测试读者对本章节内容理解了多少。
- ✧ 练习：编程练习可以提供额外的实践机会，掌握技巧和所学概念。这些练习逐渐增加难度，并且允许读者独立地了解语言和编程环境。

Web 浏览器环境

可以使用 Netscape Navigator 4 或者更高的版本、Internet Explorer 4 或者 Internet Explorer 5 来创建本书中的练习。本书中的 JavaScript 代码与 Netscape JavaScript 1.2 兼容，在 Netscape Navigator 4 或者更高版本的浏览器中支持该版本的 JavaScript。

CT 教学工具

本书中的所有教学工具都可以在教师资源工具中找到，至于教师资源工具，可以在 Course Technology 技术网站（www.course.com）或者 CD-ROM 中找到。

- ✧ 教师手册：本书的教师手册经过了严格的质量测试。手册可以在 CD-ROM 或 Course Technology 技术网站在线手册中找到。教师手册包括以下的项目：
  - 。本书中所有回顾问题、编程练习的答案和解决方案。
  - 。其中的技术提示，包括疑难点的提示。
  - 。Course 测试管理器 1.2 引擎和测试题库：Course 测试管理器（CTM）是一个基于 Windows 的有效的测试程序。该程序是专门为教学设计的，可以帮助教师设计和管理检查实际考试。这个考虑全面的程序可以让教师随意地生成实际测验的试题，并且能够立即提供在线反馈和详细答案的学习指导。教师还可以用 CTM 创建书面或通过网络生成在线测验。可以生成关于本书全部或任意章节的测验，进行预览，还可以通过局域网管理整个测验的过程。CTM 可以自动地对测验进行评分分级，还可以对读者的表现进行单独或分组的统计。在本书的 CD-ROM 中包含一个 CTM 的测验题库。测试题库包括多项选择题、判断题、简答题和论述题等几种类型。
- ✧ 方案文件：方案文件包括本书中要求回答、创建或修改的问题的可能答案（由于软件发展的一些特点，读者的答案可能和本书提供的不同，但是仍然可能是正确的）。
- ✧ 数据文件：数据文件包括了读者可能会用到的关于指导和练习的所有数据，可以在 Course Technology 技术在线或者教师资源工具的 CD-ROM 中找到数据文件。还有一个包括技术提示的帮助文件用于实验室管理。

感谢

这本书代表着许多人的艰苦工作，它并不是作者一人之力可以完成的。我衷心地感谢所有在本书的出版过程中提供帮助的人。首先我要感谢责任编辑 Marilyn Freedman，感谢他的出色工作和让我成为更好的作家。我还要感谢 Nan Fritz 和她在 nSight 的职员；感谢联合出版商 Kristen Duerr；感谢出版经理 Margarita Donovan；感谢编辑 Jennifer Muroff；感谢编辑 Ellina Beletsky、Deborah Masi，还要感谢质量保证测验员 Nicole Ashton、John Freitas、Jonathan Greacen、Burt LaFountain、Brendan Taylor、Alex White。另外，我还要感谢为章节后的练习提供解决方案的 Tim Panagos；感谢哈佛大学和 RWD 技术委员会 Peggy Beckley 分会的 Mary Ann O’Brien 和 Ellie Lottero。

我还要感谢本书的审阅者们，他们提供了宝贵的意见和建议，他们是 Winona 州立大学的 Marzie Astani、Manatee 社区学院的 Raymond Calvert；Arapahoe 社区学院的 Jim Dunne、

Massachusetts 药剂和健康科学学院的 Lila Foye、Oakton 社区学院的 Katie Kalata、Bentley 学院的 Ed Kaplan、Henderson 州立大学的 Catherine Leach、Parker Compumotor 的 Le Nguyen、Dickinson 学院的 Dave Reed、Chase Bank 的 Mildred Tassone、Gettysburg 学院的 Rod Tosten、Penn 学院的 Lara Van Nostrand。

同样，我还要感谢我的朋友和同事——George T.Lynch，是他促使我开始了本书的工作。更重要的是，我要感谢我出色的妻子 Kathy，是她给予我无比的耐心。我还要感谢一直陪伴我的猫 Mabeline，还有爱犬 Noah，是它陪我出去散步，让我从计算机前面解放出来。我还要感谢我的家人和朋友，是他们在我感觉失望的日子里给了我理解。最后，谨以本书献给我的父亲。

Don Gosselin

目　　录

第 1 章　JavaScript 简介

案例

WebAdventure Inc.是一个 Web 站点设计公司，您刚被聘为该公司的项目经理。像现在很多人那样，您可能会为能够在一个 Web 站点设计公司工作，而感到非常兴奋，每天都花费大量的时间在 Internet 上冲浪。Web 页包含有使用不同方式格式化的不同类型的信息。可能也有些 Web 站点比其他的网站的交互性更好，而且还有很多几年前所没有的新特征。游戏、订单、动画以及各种不同的视觉效果，现在几乎已经成为您所看到的每一个 Web 页的组成部分。您此时会对学习如何开发这些类型的网页感到非常兴奋。您在 WebAdventure 面临的第一个挑战是对 Web 开发和程序设计的理解，以及了解 JavaScript 的用途。

1.1　程序设计、HTML 和 JavaScript

本节目标

在本节您会学到：

- ✧ 关于万维网（World Wide Web）
- ✧ JavaScript 的用处
- ✧ 关于超文本标记语言（HTML，Hypertext Markup Language）
- ✧ 如何创建 HTML 文档
- ✧ 关于 JavaScript 程序设计语言
- ✧ 关于逻辑与调试

1.1.1　万维网

JavaScript 存在于万维网的 Web 页中，所以了解一些有关万维网的常识对理解 JavaScript 非常有帮助。

1989 年，万维网（Web）创建于瑞士日内瓦的欧洲量子物理实验室，最初是为了轻松地访问 Internet 中存在的交叉引用文档。文档的定位和打开都使用超文本链接，超文本链接

中包含有指定文档的一个引用。超文本标记语言（HTML）是一种非常简单的、用于设计万维网 Web 页的语言。Web 浏览器是一个在用户计算机上显示 HTML 文档的程序。目前，最流行的两种 Web 浏览器是 Netscape Navigator 和 Microsoft Internet Explorer。

提示：在本书中 HTML 文档和 Web 页面可以互换使用。

每一个 Web 页或文档都有一个被称为统一资源定位符（Uniform Resource Locator，简称 URL）的惟一地址。可以把 URL 想象为一个 Web 页的电话号码。每一个 URL 包括四个部分：协议（通常为 HTTP）、服务、Web 服务器的域名或 Internet 协议地址（IP 地址）以及文件名。超文本传输协议（Hypertext Transfer Protocol，简称 HTTP）负责管理用于 Web 导航的超文本链接，可以认为 Web 是靠 HTTP 驱动的。HTTP 确保 Web 浏览器能够正确地处理和显示 Web 页面中所包含的不同类型的信息（文本、图形以及其他信息）。URL 的协议部分后面紧接着的是一个冒号和两个斜杠，然后是服务，在万维网上服务通常是 www，代表了“ World Wide Web ”。域名是用来在 Internet 上标识计算机的惟一地址，这些计算机通常是 Web 服务器。域名包括两个部分，使用点号分开。域名的前一部分通常由一段简单的、标识一个人或一个组织的文本组成，例如 DonGosselin、Course。域名的后一部分则表示该站点的类型，例如 com（company 的简写）表示私营公司，gov（government 的简写）表示政府机构，edu（educational 的简写）表示教育部门。举一个例子，course.com 就是 Course Technology 的域名。像 http://www.DonGosselin.com 和 http://www.course.com 就是两个完整 URL 的例子。

提示：Internet 协议地址，或 IP 地址，由一串分为四组的数字组成，是用来惟一标识连接到 Internet 上的计算机或设备的另外一种方法。所有的 Internet 域名都对应一个惟一的 IP 地址。

在 URL 中，域名或 IP 地址后面可以带有一个指定的文件名，或者目录与文件名的组合。如果 URL 中没有指定文件名，处理请求的 Web 服务器就会在根目录或者指定目录中查找一个名为 index.html 的文件。图 1-1 中指出了一个 URL 的各组成部分，该 URL 用来打开一个名为 html_ref.html 的 HTML 文档。

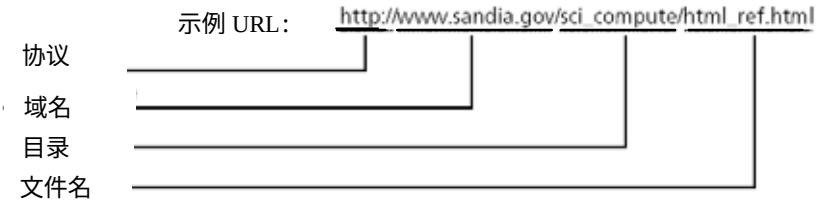


图 1-1　URL 示例

1.1.2　JavaScript 在 Web 上的作用

万维网最初的目的是为了定位和显示信息。当 Web 发展到超出学院和科学团体的范围

时，人们逐渐认识到更好的交互性会使 Web 更加有用。随着 Web 商业应用的发展，提高 Web 站点交互性和可视性的要求也随之增长。但是，使用基本 HTML 创建的文档是静态的，因为 HTML 的主要目的就是告诉浏览器如何显示文档。HTML 文档实际上就相当于使用字处理器或者桌面出版程序创建的文档——您只能浏览或者打印。为了满足提供更好的交互性的要求，Netscape 公司开发了 Navigator 浏览器中使用的 JavaScript 程序设计语言。

JavaScript 使 HTML 面貌焕然一新，使 Web 页面具有了动态特征。与静态的 HTML 文档相比，JavaScript 可以把 HTML 变成应用程序，例如游戏或订单。使用 JavaScript 可以在浏览器显示 Web 页面之后改变内容，可以通过表单或控制提供用户交互操作，创建动画之类的可视化效果，甚至控制 Web 浏览器窗口本身。这些操作在 JavaScript 问世之前是不可能的。

Web 现在有多种不同用途，例如广告和娱乐，相当一部分要归功于 JavaScript。今天，许多公司都有 Web 站点（也许以后会更多）。为了吸引更多的人来访问网站，并且把他们留在这里，公司的 Web 站点必须是吸引人的、可交互的，并且能够在感官上给人以享受。为了销售他们的产品，商业 Web 站点使用了特别的广告、动画，交互性、直观的导航控制以及其他许多类型的特殊效果。所有这些类型的应用程序和效果都可以使用 JavaScript 实现。

请看图 1-2、图 1-3 和图 1-4，这三个示例对了解 JavaScript 的用途有所帮助。在本书稍后的教程中，可以创建这些程序。图 1-2 中显示的地图是一个图像映射。当移动鼠标经过一个国家时，该国家的地图会高亮度显示，而且在 Web 页面的底端会显示这个国家的名字。第 2 章中会讲到创建图像映射。图 1-3 中显示的在线计算器会在第 3 章中讲到。在图 1-4 中显示的最后一个程序是一个在线产品注册表单，会在第 6 章中讲到。这些程序仅仅是在学习本书过程中使用 JavaScript 创建程序的一小部分。

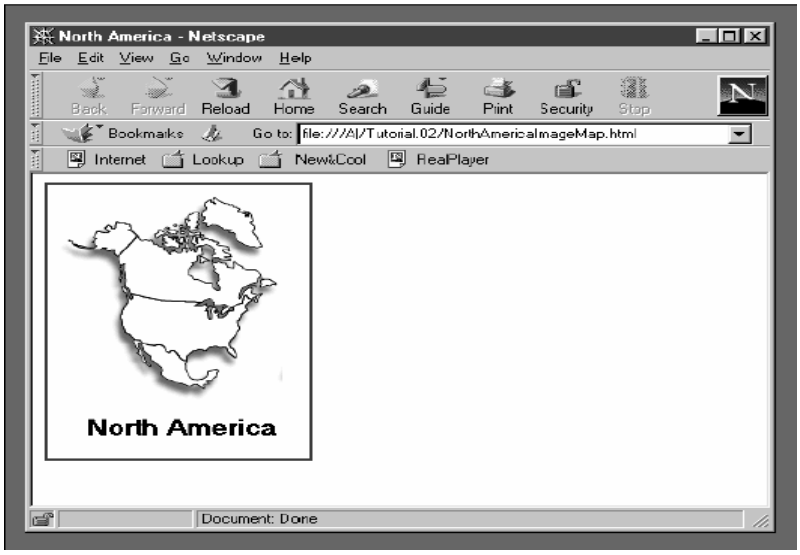


图 1-2　图像映射

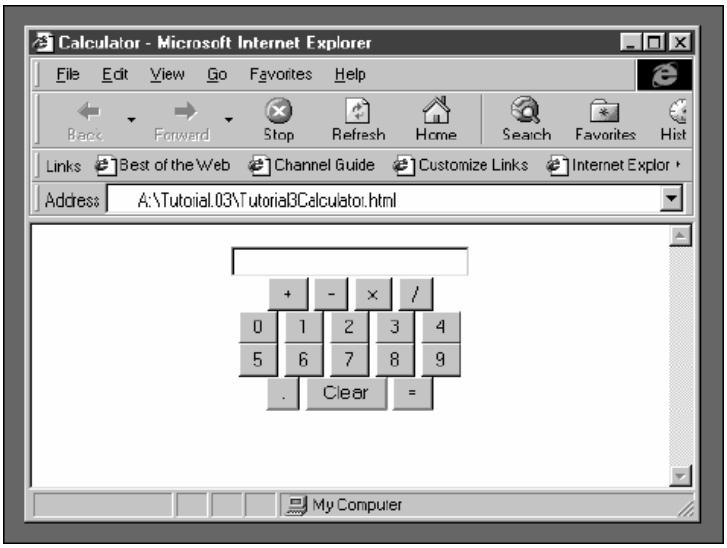


图 1-3　在线计算器

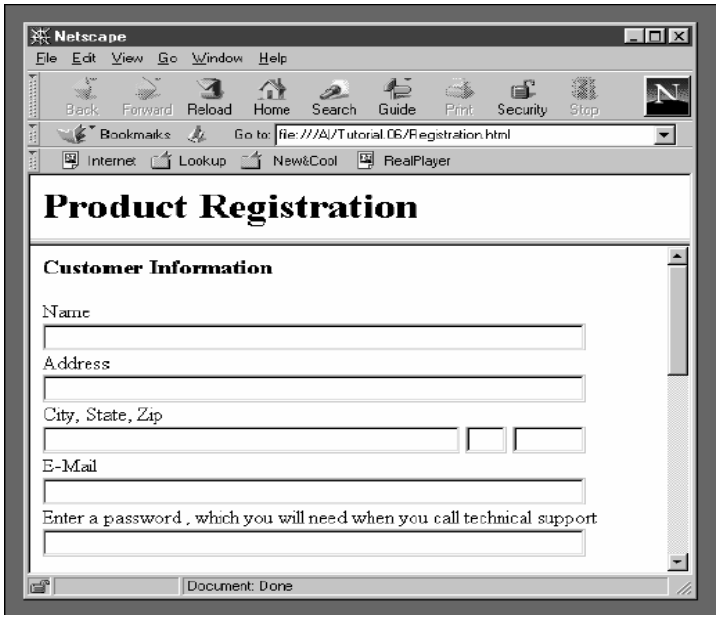


图 1-4　在线产品注册表单

1.1.3　超文本标记语言

因为 JavaScript 存在于 Web 页之中，所以要想使用 JavaScript，首先必须理解超文本标记语言和如何构造 Web 页面。本节讲解了使用 JavaScript 必须知道的 HTML 基本规则。在本书中，还会讲述一些其他必需的 HTML 知识。如果已经熟练掌握了 HTML，可以跳过

本节。

HTML 必须是文本文档，由被称为标签的格式指示以及要在 Web 页上显示的文本组成。HTML 标签的范围从格式化命令（例如黑体和斜体文本）到用户输入的控制（例如单选按钮和检查框）。还有一些 HTML 标签用于在文档或 Web 页中显示图形图像与其他对象。

提示：HTML 文档必须以.html 或.htm 作为文件扩展名。

在 Web 浏览器中打开一个 HTML 文档时，就会根据文档中的标签进行汇编和格式化。Web 浏览器汇编和格式化 HTML 文档的过程通常称为解析或翻译。标签包含在尖扩号中（<>），大多数情况下有一个起始标签和结束标签，中间是文本或者其他一些用于格式化和控制的条目。例如，一行粗体字的起始标签是<B>，结束标签是</B>。在 Web 浏览器中打开 HTML 文档时，位于这对标签中的所有文本都会显示为粗体。下面就是一个如何在 HTML 中使用粗体字的例子：

<B>This text will be boldfaced in a Web browser.</B>

提示：HTML 对大小写不敏感，所以可以使用<b>代替<B>。但是，使用大写字母作为 HTML 标签是一种习惯。

所有的 HTML 文档都以<HTML>开始，以</HTML>结束。这对标签告诉浏览器，它们之间的所有文本都应该被编译为一个 HTML 文档。表示开始和结束的<HTML>…</HTML>标签是必需的，而且把所有文本以及其他用于编排 HTML 文档的标签包含于其中。HTML 提供了许多用于创建 HTML 文档的标签。表 1-1 中列出了一些常用标签。

也可以使用不同的参数设定许多 HTML 标签，这些参数通常被称为属性。属性位于起始标签的结束尖括号之前。例如，用于在 HTML 文档中嵌入图像和视频的<IMG>标签就可以使用许多参数进行设定，包括 SRC 属性，该属性用于指定图像文件或视频剪辑的文件名。要在<IMG>标签中包含 SRC 属性，敲入<IMG SRC=＂mygraphic.gif＂>。必须用双引号把参数值括起来。

表 1-1　常用 HTML 标签

HTML 标签	描　　述
<B></B>	用粗字体格式化文本
<BODY></BODY>	封装 HTML 文档体
 	插入换行符
<CENTER></CENTER>	根据页面宽度在浏览器内居中显示文本
<HEAD></HEAD>	封装页首，包含关于整个页面的信息
<H*n*></H*n*>	标题层次标签，*n* 表示从 1 到 6 的数字
<HR>	插入水平线
<HTML></HTML>	表示 HTML 开始和结束的标签
<I></I>	用斜体字格式化文本
<IMG src="…">	插入图像文件

续表

HTML 标签	描　　述
<P>	开始一个新段落
<STRONG></STRONG>	使用 Strong 字体格式化文本，类似于粗体字
<TABLE></TABLE>	创建一个表格。<TR>定义行，<TD>定义单元格
<TITLE></TITLE>	封装页面标题，页面标题就是显示于浏览器标题栏的文本——这对标签必须出现在<HEAD>标签之中
<U></U>	封装下划线字体

<HEAD>和<BODY>是两个重要的标签。<HEAD>标签包含了提供给 Web 浏览器使用的信息，它位于 HTML 文档的开头处，在起始<HTML>标签之后。有几个标签可以放在<HEAD>…</HEAD>标签对之中，帮助组织文档内容。<TITLE>标签用于封装显示于浏览器标题栏的文本，它是<HEAD>标签惟一必需的元素。除<TITLE>标签之外，<HEAD>标签中的其他元素并不会影响 HTML 文档的翻译。HTML 文档中包含有 JavaScript 程序时，它们通常放在<HEAD>…</HEAD>标签之间。表 1-2 列出了<HEAD>…</HEAD>标签之间可以放置的标签。

表 1-2　<HEAD>…</HEAD>标签之间可以放置的 HTML 标签

HTML 标签	描　　述
<TITLE>…</TITLE>	包含文档标题，用于浏览器标题栏显示
<STYLE>…</STYLE>	标示样式表
<LINK>	指定一个 HTML 文档与外部资源的外部链接
<SCRIPT>…</SCRIPT>	声明嵌入脚本
<ISINDEX>	创建单字段搜索表单
<BASE>	标示文档 URL 基地址
<META>	包含文档属性

<HEAD>标签之后是<BODY>标签，它包含了 HTML 页面体。<BODY>标签的属性决定了 HTML 文档的外观。表 1-3 中列出了<BODY>标签的属性。

表 1-3　<BODY>标签属性

属　　性	描　　述
ALINK	激活链接的颜色
BACKGROUND	背景图像
BGCOLOR	背景颜色
LINK	未访问链接的颜色
TEXT	文本颜色
VLINK	已访问链接的颜色

Web 浏览器解析或翻译 HTML 文档时，会忽略代码中的非打印字符，例如空格、制表符和回车换行。只有最终文档中包含的可识别的 HTML 标签和文本会显示在浏览器中。不能使用回车换行在 HTML 文档的段前或段后插入空格，浏览器只把<P>和
识别为回车换行。如果使用段落标签<P>在 HTML 文档中创建空行，需要对特定的 Web 浏览器加以注意，例如 Internet Explorer，因为它会忽略无内容的标签。如果使用<P>标签在 HTML 文档中创建空行，Web 浏览器可能会完全忽略它。为了防止这种情况发生，可以在标签之间插入不可分空格代码 ;并且加上一个结束标签，完整的段落标签应该是<P> ;</P>。图 1-5 显示了一个 HTML 文档内容，图 1-6 是它在 Web 浏览器中的显示效果。

```
<HTML>
<HEAD>
<TITLE>Hello World</TITLE>
</HEAD>
<BODY>
<H1>Hello World (this is the H1 tag)</H1>
<H2>This line is formatted with the H2 tag</H2>
<P>This body text line contains several character formatting
tags including <I>italics</I>, <B>bold</B>, <U>underline</U>,
and <STRIKE>strikethrough</STRIKE>. The following code line
creates a line break followed by a horizontal rule.<BR>
<HR>
<IMG src="Checkmrk.jpg">This line contains an image.
</BODY>
</HTML>
```

图 1-5　一个 HTML 文档

提示：某些 HTML 标签，例如段落标签<P>，并不需要结束标签。另外，不同的浏览器对何时需要结束标签有不同的要求。

1.1.4　创建一个 HTML 文档

因为 HTML 文档是文本文件，所以可以在任何文本编辑器中创建它，例如记事本、写字板或者任何可以创建简单文本文件的字处理程序。如果使用文本编辑器创建 HTML 文档，

只有在 Web 浏览器中打开此文档才能够看到最终结果。不过，现在有许多专为创建 HTML 文档而设计的应用程序（称为 HTML 编辑器）。一些常用的 HTML 编辑器，例如 Microsoft FrontPage 和 Adobe PageMill，都采用图形界面，允许用户直接创建 Web 页面，而且可以立即看到结果，这有些类似于字处理器中的所见即所得（what you see is what you get，缩写为 WYSIWYG）特征。另外，现在有许多字处理应用程序，例如 Microsoft Word 和 WordPerfect，都允许用户把文件保存为 HTML 文档。所有这些编辑器创建的仍然是简单的文本文件，不过它们自动化了标签应用过程。例如，如果在 Word 中创建的文档中包括粗体字，当把它保存为 HTML 文档时，粗体标签<B>会自动加在所创建的 HTML 文本文件里的文本之前。

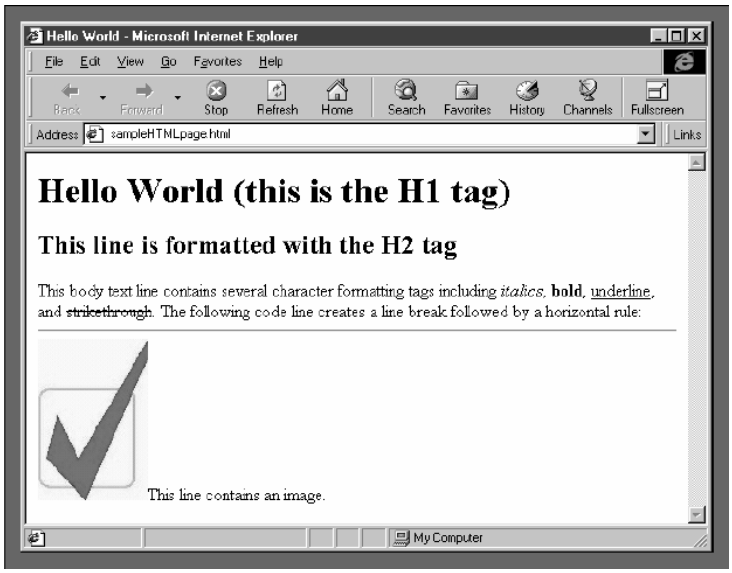


图 1-6 Web 浏览器中的 HTML 文档

提示：注意不同的浏览器翻译 HTML 文档的方式有所不同。例如，在 Internet Explorer 和 Netscape Navigator 中显示同一个 HTML 文档效果可能会不同。

接下来，创建一个 HTML 文档，其中包含了本节中所见到的一些标签。可以使用任何一个文本编辑器，例如记事本、写字板或 HTML 编辑器。

创建一个 HTML 文档的步骤：

1. 启动文本编辑器或 HTML 编辑器，创建一个新文档。

2. 输入<HTML>开始一个 HTML 文档。记住所有的 HTML 文档都应该以<HTML>…</HTML>标签对作为开始和结束。

3. 回车，在文档中紧接着输入<HEAD>和<TITLE>。标题会显示在 Web 浏览器的标题栏中。记住<HEAD>…</HEAD>标签对中必须包括<TITLE>…</TITLE>标签对。<TITLE>…</TITLE>标签对不能放在<HEAD>…</HEAD>标签对之外。

```
<HEAD>
```

```
<TITLE>Web Page Example</TITLE>
</HEAD>
```

4. 回车，输入<BODY>开始 HTML 文档的文档体部分。

5. 回车，输入下列标签和文本创建 HTML 文档的文档体。

```
<H1>Hello World</H1>
<P>This is my first Web page.</P>
<HR>
<H2>This line is H2</H2>
<H3>This line is H3</H3>
<P>The following line is empty.</P>
<P> </P>
<P><B>bold</B>,<I>italic</I>,<U>underline</U></P>
```

6. 回车，输入标签对<BODY>…</BODY>和<HTML>…</HTML>的标签结束，并结束文档的编辑。

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

7. 把文件保存为盘上 Tutorial.01 目录下的 HelloWorld.html 文件。

提示：某些 Web 服务器不能正确解释 HTML 文件名中的空格。例如，文件名为 Hello World.html（Hello 和 World 中间有一个空格）在某些服务器上可能不能正确解释。由于这个原因，本书中的文件名都不包含空格。

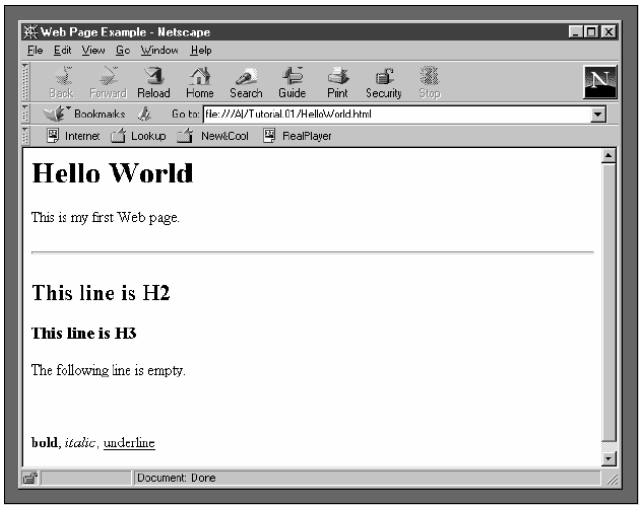


图 1-7　Navigator 4.0 中的 HelloWorld.html

8. 启动 Navigator、Internet Explorer 或其他 Web 浏览器。然后在 Web 浏览器中打开 HelloWorld.html 文件。图 1-7 中显示了在 Navigator 4.0 中打开 HelloWorld.html 文件的情况。

9. 单击“关闭”按钮结束 Web 浏览器的运行。

1.1.5 JavaScript 程序设计语言

JavaScript 是一种脚本语言。术语脚本语言是指在 Web 浏览器内由解释器解释执行的编程语言。每次运行程序的时候，解释器都会把程序代码翻译为可执行格式——每次一行。使用脚本语言编写的程序，例如 JavaScript，都是在脚本引擎装载 HTML 页面时解释执行的。脚本引擎是一个解释器，它是 Web 浏览器的一部分。一个包含翻译脚本的脚本引擎的 Web 浏览器称为脚本宿主。Navigator 和 Internet Explorer 都是 JavaScript 程序的脚本宿主的范例。

JavaScript 语言首次出现在 Navigator 中时被称为 LiveScript。随着 Navigator 2.0 的发行，该语言改称为 JavaScript 1.0。随后，Microsoft 也发行了自己的用于 Internet Explorer 的 JavaScript 语言，并称之为 JScript。目前最新版本的实现是 Navigator 中的 JavaScript 1.3 和 Internet Explorer 中的 JScript 3.0。然而，这两种实现之间并不是完全兼容。本书的目的是创建在 Navigator 和 Internet Explorer 中都可以运行的 JavaScript 程序。因此，本书的重点放在与 Netscape JavaScript 1.2 兼容的 JavaScript 代码上，因为 Netscape 4 和 Internet Explorer 4 都支持它。Netscape JavaScript 1.2 被视为目前的 JavaScript 标准，而且与 JScript 2.0 基本兼容。

提示：欧洲计算机制造商联合会，即 ECMA，创造了一个国际通用的标准化版本的 JavaScript，被称为 ECMAScript。ECMAScript 与 JavaScript 1.1 基本兼容。未来的 Netscape JavaScript 版本预计会完全符合 ECMAScript 标准。Microsoft 也宣称 JScript 3.0 与 ECMAScript 100%兼容。

提示：许多人认为 JavaScript 与 Java 编程语言相关，或者是它的一个简化版本。但是，它们是完全不同的语言。Java 是由 Sun Microsystems 公司创造的一种编译的、面向对象的程序语言，掌握 Java 要比 JavaScript 难得多。尽管 Java 常用来创建运行于 Web 页面之中的程序，但是 Java 是独立于浏览器执行的外部程序。相反的是，JavaScript 运行在 Web 之中，而且可以控制浏览器。

JavaScript 有两种形式可用：客户端 JavaScript 和服务器端 JavaScript。标准化的客户端 JavaScript 可被用于在 Web 浏览器（客户端）显示的 HTML 页面中。Navigator 4.0 中的 JavaScript 1.2 和 ECMAScript 都是 JavaScript 的客户端版本。服务器端 JavaScript 是被 Web 服务器用来访问文件系统、与其他应用程序通信、访问数据库和执行其他任务的。目前，服务器端 JavaScript 都是专有的、与供应商相关的。用户必须知道每种 Web 服务器所用语言的细微差异——还没有类似于 ECMAScript 的服务器端标准。客户端和服务器端 JavaScript

都具有相同的基本编程语言特征。图 1-8 描述了客户端和服务器端 JavaScript 的关系。

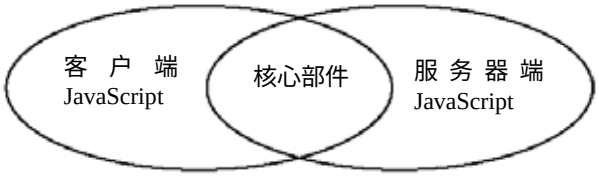


图 1-8　客户端和服务器端 JavaScript 的关系

1.1.6　逻辑与调试

所有的程序设计语言，包括 JavaScript，都有自己的语法或语言规则。编写程序时，必须理解给定程序设计语言的语法，而且还必须理解计算机程序设计逻辑。作为所有程序基础的编程逻辑包括采用正确的顺序执行程序的不同部分以获得所需的结果。例如，尽管您很清楚地知道如何驾驶一辆汽车，但是如果不按照正确的路线行驶也不会到达目的地。类似的，您可能能够正确地使用一种程序设计语言的语法，但是却不能运行一个合乎逻辑构造的、可以工作的程序。逻辑错误的例子包括在对两个数做除法时却做了乘法，在获得正确的输入之前产生输出。下面这段 JavaScript 代码中包含了另外一个逻辑错误的例子。

```
var count = 1
while (count <= 10) {
        alert("The number is " + count);
}
```

上面这段代码中使用了一个 while 语句，它根据对特定条件的判断而重复执行一条或一系列命令。本例中 while 语句的条件是变量 count 的值（变量在第 2 章中讲述）在小于或等于 10 的情况下 alert(…)应该一直被执行。然而，while 语句体内却没有改变变量 count 值的代码。所以在每一次循环后，count 变量的值还是 1。在这种情况下，无论按多少次“ OK ”按钮，包含“ The number is 1 ”字符串的报警对话框还是会一遍又一遍地被显示。这种类型的逻辑错误称为死循环。

提示：不要担心例子中的 JavaScript 代码是如何构成的。例子的目的仅仅是让您对逻辑错误有一个更好的理解。

不管是因为语法错误还是逻辑缺陷，程序中任何造成功能不正确的错误都被称为一个程序故障（Bug）。调试（Debugging）指的就是跟踪和解决程序中错误的活动。调试这个术语第一次由 Grace Murray Hopper 提出时有一个典故。Hopper 是一位开发 COBOL 语言的数学家，在故事发生的 1940 年，Hopper 使用的一台旧式计算机由于虫蛀而短路。把蛀虫从计算机中清除后排除了计算机的故障，解决了麻烦。今天，程序故障指的是在程序设计

或操作中出现的各种错误。

提示：不要混淆程序故障和计算机病毒。程序故障是指由于语法错误、设计缺陷或运行时错误而导致程序发生的问题。病毒是一个完备的程序，设计用来“感染”计算机系统或造成善意或恶意的破坏。实际上，如果有语法错误或没有执行创建者所设想的操作（或造成破坏），病毒程序自身可能也包含程序故障。

许多程序设计语言中都包括命令和其他用于辅助程序故障定位的功能。然而，本书中涉及的 JavaScript 版本却并没有任何有用的调试工具。JavaScript 以后的版本可能会包含调试功能。本书会在提示中介绍一些调试技巧和调试建议。但是，当阅读调试技巧的时候，必须清楚地认识到调试并不是一门非常精确的科学——您所写的每一个程序都是不同的，都需要使用不同的调试方法。您自己的逻辑和分析技巧是您拥有的最好的调试资源。

在下一节，就会开始学习如何创建 JavaScript 程序。

1.1.7 总结

- ✧ 超文本标记语言是一个用于万维网 Web 页设计的简单协议。
- ✧ 万维网由超文本传输协议（HTTP）驱动，它用于管理负责 Web 导航的超文本链接。
- ✧ 每一个 Web 文档都有一个被称为统一资源定位符（URL）的惟一地址。
- ✧ JavaScript 使 HTML 变得焕然一新，使 Web 页面具有了动态特征。JavaScript 可以把 HTML 变成应用程序，例如游戏、订单。
- ✧ HTML 必须是文本文档，由被称为标签（Tags）的格式指示以及要在 Web 页上显示的文本组成。
- ✧ HTML 标签的范围从格式化命令到允许用户输入的控制，以及显示图形图像与其他对象。
- ✧ Web 浏览器汇编和格式化 HTML 文档的过程通常称为解析或翻译。
- ✧ 每次运行程序的时候，解释器都会把程序代码翻译为可执行格式——每次一行。
- ✧ JavaScript 是一种解释型程序设计语言。
- ✧ 脚本引擎是一个解释器，它是 Web 浏览器的一部分。一个包含翻译脚本的脚本引擎的 Web 浏览器称为脚本宿主。
- ✧ 标准化的客户端 JavaScript 可被用于由 Web 浏览器显示的 HTML 页面中。
- ✧ 所有的程序设计语言，包括 JavaScript，都有自己的语法或语言规则。
- ✧ 作为所有程序基础的编程逻辑包括采用正确的顺序执行程序的不同部分以获得所需的结果。
- ✧ 调试指的就是跟踪和解决程序中错误的活动。

1.1.8　问题

1．_____管理用于 Web 导航的超文本链接。

a. Netscape Navigator
b. 欧洲量子物理实验室
c. 超文本传输协议
d. Internet Explorer

2．每一个 Web 文档都有一个被称为_____的惟一地址。

a. IP 地址
b. 超链接
c. 统一资源定位符
d. 域名

3．HTML 元素_____。

a. 必须包括起始和结束标签
b. 仅包括起始标签
c. 可能包括结束标签，根据不同的 HTML 元素而定
d. 不包括起始和结束标签

4．HTML 是_____。

a. 对大小写敏感的
b. 对大小写不敏感的
c. 首字母必须大写
d. 必须使用大写字母

5．HTML 文档以_____标签对开始和结束。

a. <BODY>…</BODY>
b. <HEAD>…</HEAD>
c. <HTML>…</HTML>
d. <WEB>…</WEB>

6．HTML 属性放在_____。

a. 开始标签的开始括号内
b. 结束标签的开始括号内
c. 开始标签的结束括号内
d. 结束标签的结束括号内

7．程序设计语言的规则被称为_____。

a. 过程
b. 汇编
c. 语法

d. 逻辑

8．采用正确的顺序执行程序的不同语句和过程获得所需结果是指_____。

a. 推理

b. 定向汇编

c. 语法

d. 逻辑

9．术语_____语言是指在 Web 浏览器内部运行的解释语言。

a. Internet 程序设计

b. 机器

c. 汇编

d. 脚本

10．Web 浏览器中 HTML 页面可用的 JavaScript 版本被称为_____JavaScript。

a. 预编译

b. 服务端

c. 嵌入式

d. 客户端

1.1.9 练习

1．设计一个作为销售体育用品公司主页的 HTML 文档。如果能访问剪辑库，请在文档中包含公司销售的运动器材的图片，例如篮球、棒球手套、网球拍等。针对每一种商品，请包含制造商和销售价格等信息。文档中应该包括<HEAD>和<BODY>段。在<HEAD>段包含一个<TITLE>，至少要使用五种不同的 HTML 元素来格式化<BODY>段。把这个 HTML 文档保存为 SportsCompany.html，并放在盘上的 Tutorial.01 目录中。尽管您只能使用静态 HTML 元素创建这个 Web 页（这就是说，还不能在文档中包含任何动态的 JavaScript 元素，例如订货单），但是可以先列出自己想要添加到 Web 页中的动态元素，例如商品目录申清单。

2．使用 Yahoo 或其他搜索引擎搜索有关万维网历史的信息，用一页纸写一篇文章讲述一下有什么收获。

3．访问万维网协会（W3C）的网址 http://www.w3c.org，并阅读一下有关最新的 HTML 规范的内容。针对 HTML 的改进写一个摘要，并说明这些改进将会如何影响 Web 页设计。

4．在 Web 上搜索三个 JavaScript 编写的完整程序或 JavaScript 作用的例子。用一页纸描述一下这些程序或作用，并说明您是否觉得它们有效果。

5．Jakob Nielsen 创作了一个名为 AlertBox、大受欢迎的 Internet 专栏，这个专栏的网址是 http://www.useit.com/alertbox。阅读一下 Alertbox 中的以下三个专栏：“The Top Ten Mistakes in Web Design”（May 30,1999）、“Who commits the Top Ten Mistakes in Web Design?”（May 16,1999）和“Top Ten Mistakes’ revisited three years later”（May 2,1999），

并对您从中所得写一个短评。

6．在图书馆或书店，或者 Internet 上查找有关编程逻辑和调试的信息。用一页纸写一下您的收获。

7．在 Internet 搜索有关 Internet 协议、域名和 IP 地址的信息，并写一个简短的摘要，阐述一下它们彼此之间如何互相作用。

1.2　第一个 JavaScript 程序

本节目标

在本节将学习：

- ✧ 关于<SCRIPT>标签
- ✧ 如何创建 JavaScript 源代码文件
- ✧ 如何为 JavaScript 程序添加注释
- ✧ 如何创建 HTML 文档
- ✧ 如何为不兼容的浏览器隐藏 JavaScript 代码
- ✧ 关于在 HTML 文档的<HEAD>或<BODY>段放置 JavaScript 代码

1.2.1　关于<SCRIPT>标签

JavaScript 程序在 HTML 文档中运行，HTML 文档中组成 JavaScript 程序的语句包含在<SCRIPT>…</SCRIPT>标签对中。<SCRIPT>标签用来通知 Web 浏览器随后的指令由脚本引擎来解释。<SCRIPT>标签的 LANGUAGE 属性告诉浏览器使用的是哪种脚本语言和脚本语言的版本。在 HTML 文档中包含下面的代码，表示接下来的语句应该由 JavaScript 脚本引擎来解释：

```
<SCRIPT LANGUAGE="JavaScript">
JavaScript 语句
</SCRIPT>
```

提示：尽管本书中讲的是 JavaScript，但是在 Web 页中也可以使用其他类型的脚本语言。Microsoft 的 VBScript 就是另外一种类型的脚本语言，它基于 Visual Basic 程序设计语言。如果在 HTML 中使用 VBScript，应该使用<SCRIPT LANGUAGE="VBScript">VBScript 语句</SCRIPT>。请不要混淆 JScript 和 VBScript，JScript 是 Micriosoft 公司的 JavaScript 脚本语言版本。为了指定为 JScript 语言，应该在 LANGUAGE 属性中指明为 JavaScript。

JavaScript 是大多数 Web 浏览器的默认脚本语言。如果忽略了<SCRIPT>标签中的 LANGUAGE 属性，JavaScript 仍然可以运行。然而，Internet 总是在改变的。新技术，包括新的脚本语言，总是不断推出。很难判断像 VBScript 这样有竞争力的脚本语言以后是否会占优势。所以，使用<SCRIPT>的 LANGUAGE 属性告诉浏览器使用何种脚本语言总是明智的选择。

也可以使用 LANGUAGE 属性指明所使用的 JavaScript 的版本。特定的 Web 浏览器只支持特定版本的 JavaScript。例如，Navigator 3.0 只支持 JavaScript 1.1 及其以下版本。当使用<SCRIPT>标签中的 LANGUAGE 属性指明 JavaScript 的版本时，应该删除 JavaScript 和正确的版本号之间的空格。如果在脚本语言名称和版本号之间包含空格的话，浏览器就不会解释代码。Web 浏览器不能解释 JavaScript 1.1，因为在 JavaScript 和版本号之间有一个空格。下面的代码指明所使用的 JavaScript 代码与 JavaScript 1.1 及其以下版本兼容。

```
<SCRIPT LANGUAGE="JavaScript1.1">
JavaScript 语句
</SCRIPT>
```

表 1-4 中列出了不同版本的 Navigator 和它们所支持的 JavaScript 的版本，以及在<SCRIPT>标签中应该包含的正确的代码。

表 1-4 Navigator 支持的 JavaScript 版本

Netscape 版本	兼容的 JavaScript	代 码
Navigator 2.0 以前版本	不支持	
Navigator 2.0	JavaScript 1.0	<SCRIPT LANGUAGE="JavaScript1.0">…</SCRIPT>
Navigator 3.0	JavaScript 1.1 及其以下版本	<SCRIPT LANGUAGE="JavaScript1.1">…</SCRIPT>
Navigator 4.0–4.05	JavaScript 1.2 及其以下版本	<SCRIPT LANGUAGE="JavaScript1.2">…</SCRIPT>
Navigator 4.06–4.5	JavaScript 1.3 及其以下版本	<SCRIPT LANGUAGE="JavaScript1.3">…</SCRIPT>

装载一个 HTML 文档时，Web 浏览器会检查<SCRIPT>标签的 LANGUAGE 属性指明的 JavaScript 版本号。如果使用的 Web 浏览器不支持所指明的 JavaScript 版本，浏览器就会忽略<SCRIPT>...</SCRIPT>标签对中的所有语句。如果希望 Web 页能与老版本的 Web 浏览器兼容，应该指明老版本浏览器支持的 JavaScript 版本。例如，Navigator 3.0 只支持 JavaScript 1.1，所以如果想在 Navigator 3.0 版中显示 Web 页，就必须把 HTML 代码中的<SCRIPT>标签写成<SCRIPT LANGUAGE="JavaScript1.1">…</SCRIPT>。在本节后面的部

分，将学到如何把 JavaScript 代码在不兼容的浏览器中隐藏起来。

一旦通知 Web 浏览器使用的是 JavaScript 的早期版本，程序中的语法就必须和早期版本一致。本书中的代码与 Navigator 4.0 和 Internet Explorer 4.0 支持的 JavaScript 1.2 兼容。

除了是一种解释型脚本语言之外，JavaScript 还是一种面向对象的程序设计语言。对象就是能够被看作独立单元或组件的程序代码和数据。程序语言中单独一行叫做语句，与对象相联系的一组相关语句被称为方法。JavaScript 把很多东西都当作对象。JavaScript 编程中最常用的对象是 Document 对象。Document 对象代表浏览器窗口中的内容。被显示的 Web 页中的任何文本、图像或其他信息都是 Document 对象的一部分。Document 对象最常见的用法是向 Web 页中添加文本。可以使用 Document 对象的 write()方法和 writeln()方法在 Web 页添加新的文本。

执行或调用对象的方法只需在对象的后面加一个点号，然后加上所需的方法，方法的圆括号之间包含任何所需的参数。参数就是能够传递给一个方法的任何类型的信息。Document 对象的 write()和 writeln()方法需要一个文本字符串参数。文本字符串或文字串，就是包含在双引号之间的文本。作为参数传递给 Document 对象的 write()或 writeln()方法的文本就是 Document 对象用来新加到 Web 页中的文本。例如，Document.write("this is a text string");会在 HTML 文档中写入“ this is a text string ”。如果想在文字串中包含引用字符串，就需要用单引号把这段文字括起来。例如，Document.write("this is a 'text' string")会在 HTML 文档中写入“ this is a ‘text’ string ”。

write()和 writeln()方法实现的功能和在 HTML 文档中手工添加文本基本相同。无论是使用标准的 HTML 标签还是使用 write()和 writeln()方法向 HTML 文档添加文本，文本都是按照在 HTML 文件中遇到的顺序添加的。不同于标准的 HTML 文本，write()和 writeln()方法能够在浏览器显示 HTML 文档后添加新的文本。

write()和 writeln()方法之间惟一的区别就是 writeln()方法在文本行的结尾加一个回车换行符。然而，回车换行只有在 HTML 的<PRE>…</PRE>标签对之内才能被识别。<PRE>…</PRE>标签对是预格式化文本（Preformatted Text）的缩写。这个标签对告诉浏览器起始标签和结束标签中包含的任何文本和回车都准确地按照它们出现的样子显示。<PRE>…</PRE>标签对被认为是包容器元素（Container Element），因为它能够包含文本和其他 HTML 标签。为了让 Web 浏览器能够识别 writeln()方法产生的回车符，必须把<SCRIPT>…</SCRIPT>标签对用<PRE>…</PRE>标签对包围起来。

图 1-9 中包含的脚本使用 Document 对象的 writeln()方法在 Web 浏览器中打印“ Hello World ”。注意<SCRIPT>…</SCRIPT>标签对是用<PRE>…</PRE>标签对包围起来的。图 1-10 显示了输出结果。

提示：在“ HelloWorld ”脚本这样简单的 JavaScript 文件中可以忽略<HTML>、<HEAD>和<BODY>标签。

提示：虽然图 1-9 中的两个 document.writeln()语句都写在单独的一行里，但是如果语

句使用分号分开的话，可以把 JavaScript 语句写在同一行里。由于图 1-9 中的语句都写在单独的行里，所以每条语句结尾的分号其实不是必须的。但是，在语句后使用分号结尾是一个 JavaScript 编程的好习惯。

```
<PRE>
<SCRIPT LANGUAGE="JavaScript1.2">
document.writeln("Hello World");
document.writeln(
"This line is printed below the 'Hello World' line.");
</SCRIPT>
</PRE>
```

图 1-9　使用 Document 对象的 writeln()方法的“ Hello World ”脚本

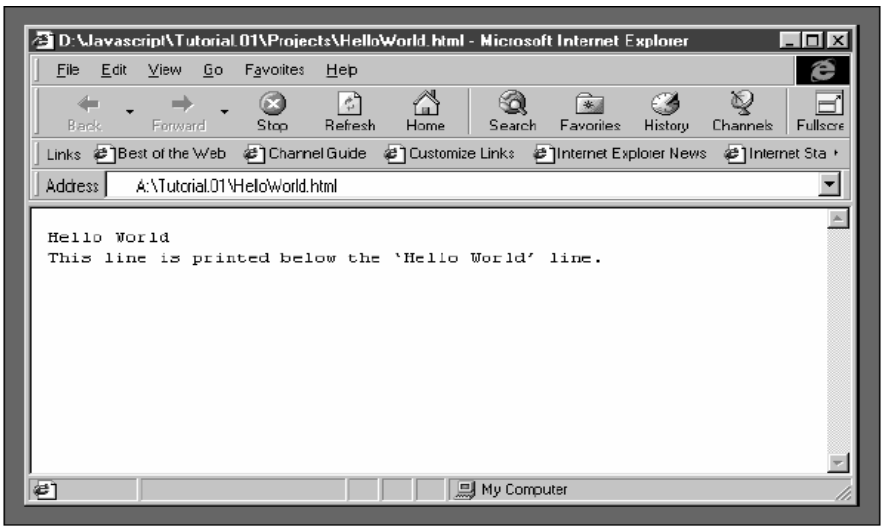


图 1-10　使用 Document 对象的 writeln()方法的“ Hello World ”脚本的输出

为了便于识别，在提到 JavaScript 程序设计语言自身所有的对象，例如 Document 对象时，使用首字母大写以把它们作为“ 顶级 ”对象。然而，不像 HTML，JavaScript 对大小写敏感。尽管在本书中我们引用 Document 对象时使用了大写的“ D ”，但是在一段脚本中引用 Document 对象必须使用小写的“ d ”。由于 JavaScript 解释器不能识别带有大写“ D ”字母的 Document 对象，所以语句 Document.write("Hello World")会产生一个错误信息。类似地，下列拼写错误的语句也会产生错误：

```
DOCUMENT.write("Hello World")
Document.Write("Hello World")
document.WRITE("Hello World")
```

图 1-11 是在 Navigator 中试图执行无效语句时显示的错误信息，图 1-12 是在 Internet Explorer 中试图执行无效语句时显示的错误信息。

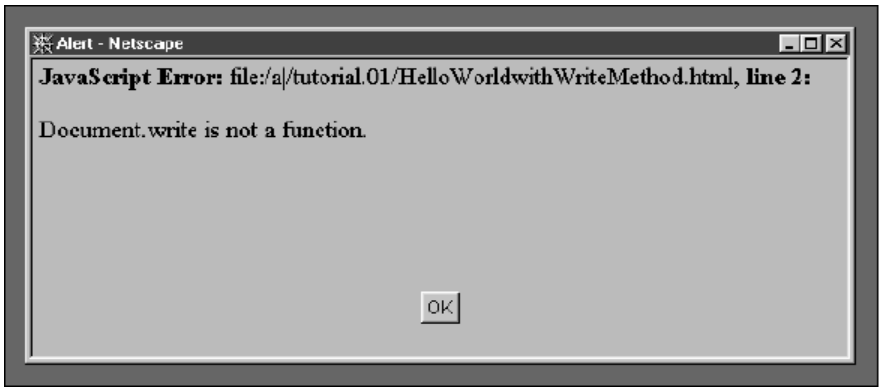


图 1-11　Navigator 中的错误信息

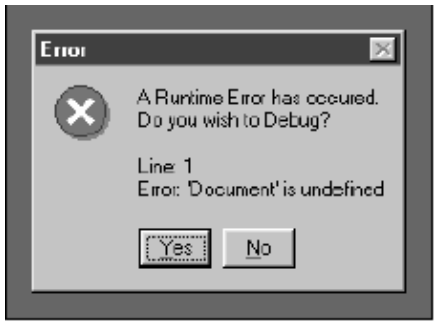


图 1-12　Internet Explorer 中的错误信息

接下来，创建一个简单的 JavaScript 文档。

创建一个 JavaScript 文档的步骤：

1. 启动文本编辑器或 HTML 编辑器，创建一个新文档。
2. 输入<PRE>开始一个预格式化文本容器。
3. 回车，输入<SCRIPT LANGUAGE="JavaScript1.2">开始 JavaScript 文档。
4. 回车，输入 document.writeln("This is the first line in my JavaScript file. ");。
5. 回车，输入 document.writeln("This is the second line in my JavaScript file. ");。
6. 回车，输入</SCRIPT>结束<SCRIPT>…</SCRIPT>标签对。
7. 输入</PRE>结束预格式化文本容器。
8. 把文件保存为盘上 Tutorial.01 目录下的 MyFirstJavaScript.html 文件。
9. 在 Web 浏览器中打开 MyFirstJavaScript.html 文件。如果收到一个如图 1-11 或图 1-12 所示的错误信息，请检查 Document.writeln()语句中对象和 writeln()方法的大小写。图 1-13 所示为 MyFirstJavaScript.html 文件在 Internet Explorer 中显示的效果。

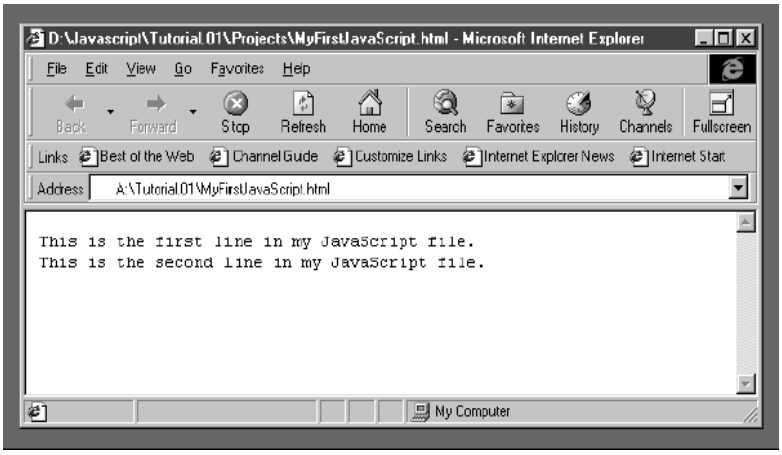


图 1-13　MyFirstJavaScript.html 文件在 Internet Explorer 中的显示

1.2.2　创建 JavaScript 源代码文件

JavaScript 通常直接插入到 HTML 文档中，但是也可以把 JavaScript 代码保存在一个被称为源代码文件的外部文件里。JavaScript 源代码文件通常使用.js 扩展文件名命名，其中仅包含有 JavaScript 语句，它不包含 HTML 中的<SCRIPT>…</SCRIPT>标签对。<SCRIPT>…</SCRIPT>标签对位于 HTML 文档中调用源代码文件的位置。访问保存在外部文件里的 JavaScript 代码需要使用<SCRIPT>标签的 SRC 属性。SRC 属性接受一个指明 JavaScript 源代码文件 URL 或目录位置的文本字符串。例如，装载位于 C:\javafiles 目录下名为 samplesourcefile.js 的 JavaScript 源代码文件，要在 HTML 中包含如下代码：

```
<SCRIPT LANGUAGE="JavaScript1.2" SRC="c:\javafiles\samplesourcefile.js">
</SCRIPT>
```

JavaScript 源代码文件中不能包含 HTML 标签。如果在 JavaScript 源代码文件中包含了 HTML 标签，就会收到一个错误信息。而且，如果在 HTML 文档中使用 SRC 属性指定了一个源代码文件，浏览器将会忽略掉位于<SCRIPT>…</SCRIPT>标签对之间所有的其他 JavaScript 代码。例如，思考一下下面的 JavaScript 代码。使用<SCRIPT>标签的 SRC 属性指定的 JavaScript 源代码文件能够被正确的执行，但是 document.writeln()语句被忽略了。

```
<SCRIPT LANGUAGE="JavaScript1.2" SRC="c:\javafiles\samplesourcefile.js">
document.writeln("this JavaScript statement will be ignored");
</SCRIPT>
```

提示：某些旧版本的 Web 浏览器，例如 Navigator 2.0，不能识别<SCRIPT>标签的 SRC 属性。

如果用在 HTML 文档中的 JavaScript 代码非常短，通常把 JavaScript 代码包含在 HTML 文档中较为简单。但是代码较长的话，把代码包含在一个.js 源代码文件里就比较方便了。使用.js 源代码文件而不是直接在 HTML 文档中添加代码的原因有以下几条：

- ✧ HTML 文档显得更干净。在 HTML 文档中使用冗长的 JavaScript 代码容易造成混淆——可能不能很快看出 HTML 代码在哪里结束，JavaScript 代码从哪里开始。
- ✧ JavaScript 代码可以被多个 HTML 文档共享。例如，Web 站点可能包含允许用户下订单的页面。每个 Web 页显示一个不同的项，但是都使用相同的 JavaScript 代码收集信息。Web 页面可以共享一个集中的 JavaScript 源代码文件，而不需在每一个 HTML 文档中重新创建 JavaScript 订单信息代码。多个 HTML 文档共享一个单独的源代码文件还可以节省磁盘空间。另外，当在多个 HTML 文档之间共享一个源代码文件时，Web 浏览器只需在内存中保留该文件的一个拷贝，减少了系统开销。
- ✧ 在不兼容的浏览器中，JavaScript 源代码文件把 JavaScript 代码隐藏起来。如果 HTML 文档中包含 JavaScript 代码，而不是调用一个外部 JavaScript 源代码文件，不兼容的浏览器就会把代码当作标准的文本显示。
- ✧ JavaScript 源代码文件可以帮助隐藏 JavaScript 代码。在花费大量时间编写完代码之后，可能不希望其他程序员拷贝代码并使用它们——尤其是在代码有版权或者您销售 JavaScript 程序解决方案时。如果您的 JavaScript 代码嵌入在 HTML 文档中，其他程序员可以通过查看 HTML 源文件的方法，轻松地拷贝您的代码，但是您的 JavaScript 代码包含在一个源代码文件中就安全些。

可以在 HTML 文档中同时使用嵌入式 JavaScript 代码和 JavaScript 源代码文件。如果有多个 HTML 文档，每个 HTML 文档都需要特殊的 JavaScript 代码，但是它们都可以共享同一个 JavaScript 源代码文件，在这种情况下，同时使用嵌入式 JavaScript 和 JavaScript 源代码文件就显得非常有利了。假设 Web 站点上有多个 Web 页面，每一页都显示公司销售的一件产品。可能有一个用于收集订单信息（例如姓名、地址等）的 JavaScript 源代码文件，所有被销售的产品都可以共享这个源代码文件。但是每一件单独的产品都可能需要使用 JavaScript 代码收集不同类型的信息。例如，产品中可能有一件衬衫，就需要收集尺寸和颜色信息。在另外一个 Web 页面里，可能销售软糖，这时就需要收集数量和口味信息。每一件产品都可以共享一个集中的源代码文件收集标准信息，同时也可以使用嵌入的 JavaScript 代码收集产品专有的信息。

在 HTML 文档中包含多个 JavaScript 代码段时，每一段都必须使用<SCRIPT>…</SCRIPT>标签对。HTML 文档中所有的 JavaScript 代码按照出现的顺序执行。图 1-14 显示了一个调用外部 JavaScript 源代码文件同时包含嵌入 JavaScript 代码的 HTML 文档。

下面这段代码执行包含在预格式化文本段的嵌入 JavaScript 代码。

```
<PRE>
<SCRIPT LANGUAGE="JavaScript1.2">
document.writeln("Your order has been confirmed.");
document.writeln("Thank you for your business.");
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

```
<HTML>
<HEAD>
<TITLE>HTML Document with Two JavaScript Sections</TITLE>
</HEAD>
<BODY>
The following two lines call an external
JavaScript source file.<BR>
<SCRIPT LANGUAGE="JavaScript1.2"
SRC="c:\javafiles\samplesourcefile.js">
</SCRIPT>
```

图 1-14　调用外部源代码文件同时包含嵌入 JavaScript 代码的 HTML 文档

下面，会创建一个调用外部 JavaScript 源代码文件而且包含嵌入 JavaScript 代码的 HTML 文档。首先创建宿主 HTML 文档。

创建宿主 HTML 文档：

1. 启动文本编辑器或 HTML 编辑器，创建一个新文档。
2. 输入起始<HTML>和<HEAD>标签。

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

3. 回车，输入标题：<TITLE>Multiple JavaScript Calls</TITLE>。
4. 回车，输入</HEAD>结束<HEAD>…</HEAD>标签对。
5. 回车，输入以下代码开始 HTML 文档体并调用一个外部 JavaScript 源代码文件。

```
<BODY>
<SCRIPT LANGUAGE="JavaScript1.2" SRC="javascriptsource.js">
</SCRIPT>
```

6. 回车，输入以下代码在预格式化容器中执行嵌入 JavaScript 代码。

```
<PRE>
```

```
<SCRIPTLANGUAGE="JavaScript1.2">
document.writeln("This line was created with embedded JavaScript code. ")
document.writeln("This line was also created with embedded JavaScript code. ")
</SCRIPT>
</PRE>
```

帮助：文字串，例如前面的 writeln()语句中的文本，必须位于同一行。如果在一个文字串中包含回车，就会收到一个错误消息。

7. 回车，输入如下代码结束<HTML>和<BODY>标签对。

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

8. 把文件保存为盘上 Tutorial.01 目录下的 MultipleJavaScriptCalls.html 文件。

接下来创建 JavaScript 源代码文件，然后打开 MultipleJavaScriptCalls.html。

创建 JavaScript 源代码文件并打开 MultipleJavaScriptCalls.html：

1. 在文本编辑器或 HTML 编辑器中创建一个新文档。

2. 输入 document.write("This line was printed from the JavaScript source file. ")。文档中只有这一行。记住在源代码文件中不包含<SCRIPT>标签。

3. 把文件保存为盘上 Tutorial.01 目录下的 JavaScriptSource.js 文件。

4. 在 Web 浏览器中打开 MultipleJavaScriptCalls.html 文件。图 1-15 所示为 Netscape 4.0 中 MultipleJavaScriptCalls.html 文件的显示效果。

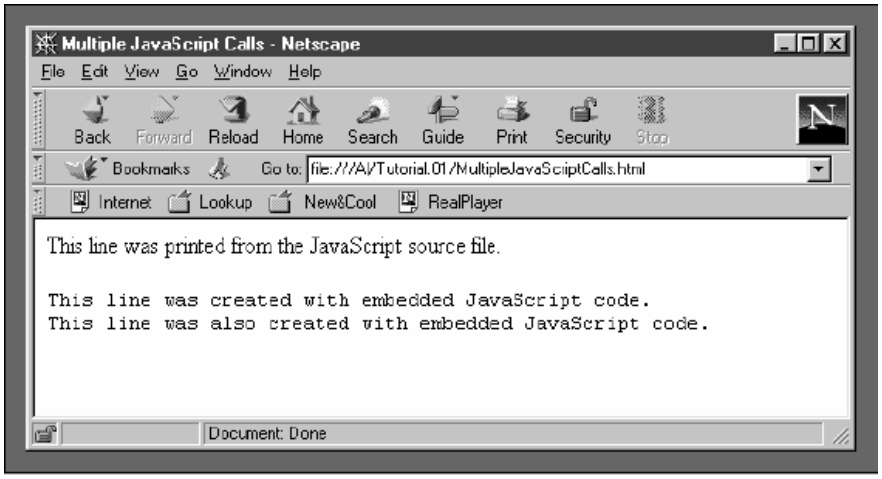


图 1-15　在 Netscape 4.0 中显示 MultipleJavaScriptCalls.html

1.2.3 为 JavaScript 程序添加注释

无论是使用 JavaScript 还是其他编程语言，在编写程序时为代码添加注释都是一种好的编程习惯。注释就是放在代码中的非打印行，其中包含各种标注，包括程序名称、姓名和创建程序的日期、给自己看的注解或者给以后可能需要修改程序的程序员的说明。当编写较长的脚本时，注释能使人更容易了解程序是如何组织的。

JavaScript 支持两种类型的注释：行注释和块注释。行注释是通过在想要用作注释的文本前加上两个斜杠//来创建。//字符指示 JavaScript 解释器忽略这一行到行尾的所有文本。行注释可以位于代码的结尾处，也可以自己组成一个完整的行。块注释可以跨越多行，通过在块中第一行前添加/*创建。可以在块中包含的最后的文本之后键入*/来结束注释块。所有位于起始的/*和结束的*/之间的文本都会被 JavaScript 解释器忽略。图 1-16 显示了一个带有行注释和块注释的 JavaScript 文件。

提示：JavaScript 中的注释和 C++与 Java 中的注释使用相同的语法。

```
<SCRIPT LANGUAGE="JavaScript1.2">
/*
This line is part of the block comment.
This line is also part of the block comment.
*/
document.writeln("Comments example"); //line comments can follow code statements
//this line comment takes up an entire line.
/*this is another way of creating block comment.*/
</SCRIPT>
```

图 1-16 带有行注释和块注释的 JavaScript 文件

接下来在 MyFirstJavaScript.html 文件中添加注释。

1. 在文本编辑器或 HTML 编辑器中打开 MyFirstJavaScript.html 文件。
2. 把光标置于包含起始<SCRIPT>标签的行尾，回车，然后添加如下注释块。

```
/*
JavaScript code for MyFirstJavaScript.html
your name
today's date
*/
```

帮助：当在 JavaScript 程序中创建注释时，一定要确保使用的是正斜杠（/）而不是反斜杠（\）。人们通常会混淆这两个字符。如果在创建注释时使用的是反斜杠而不是正斜杠，那么当试图在 Web 浏览器中打开文件时就会收到一个错误信息。

3. 把光标置于 document.writeln("This is the first line in my JavaScript file. ");语句的行尾，按 Tab 键，然后键入//Line1。

4. 把光标置于 document.writeln("This is the second line in my JavaScript file. ");语句的行尾，按 Tab 键，然后键入//Line2。

5. 保存 MyFirstJavaScript.html 文件并在 Web 浏览器中打开它，确认一下注释有没有被显示。

1.2.4　在不兼容的浏览器中隐藏 JavaScript 代码

创建 JavaScript 源代码文件可以在不兼容的浏览器中把 JavaScript 代码隐藏起来。但是，如果 HTML 文档包含嵌入 JavaScript 代码而不是调用一个外部.js 源代码文件，那么不兼容的浏览器就会把代码当作标准的文本显示出来。为了在不兼容的浏览器中隐藏嵌入的 JavaScript 代码，就需要把<SCRIPT>…</SCRIPT>标签对置于一个 HTML 注释块之中。HTML 注释不同于 JavaScript 注释。HTML 注释以<!--开始，以-->结束。所有位于注释起始标签和结束标签之间的文本都不会被浏览器提交。例如，图 1-17 显示了一个带注释的 HTML 文档。图 1-18 是显示结果。位于注释标签之间的文本没有被浏览器提交。

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>
This line is rendered normally since it is located before the opening
comment tag.<BR>
<!--Text on this line is not displayed
Text on this line is not displayed
This line is not displayed either-->
This line is rendered normally since it is located after the closing
comment tag.<BR>
</BODY>
</HTML>
```

图 1-17　带注释的 HTML 文档

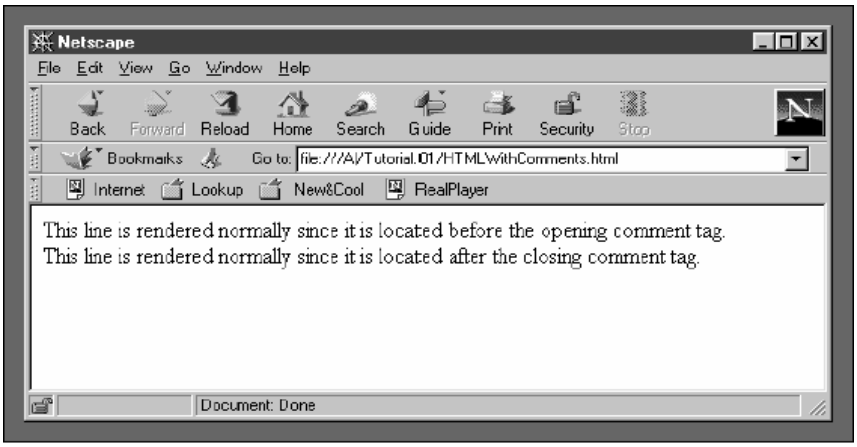


图 1-18　带注释的 HTML 文档的显示效果

大多数 Web 浏览器不会显示用 HTML 注释标签隔开的行。但是，与 JavaScript 兼容的浏览器会忽略 HTML 注释标签，正常地执行 JavaScript 代码。记住与 JavaScript 兼容的浏览器决不会显示 JavaScript 代码，而是由浏览器的脚本引擎解释。只有 JavaScript 注释标签可以把 JavaScript 代码从解释器中隐藏起来。图 1-19 显示了一个使用 HTML 注释把 JavaScript 代码从不兼容的浏览器中隐藏起来的例子，但是兼容的浏览器能执行这段代码。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!--This line starts the HTML comment block
document.writeln("Your order has been confirmed.");
document.writeln("Thank you for your business.");
This line ends ehe HTML comment block-->
</SCRIPT>
```

图 1-19　使用 HTML 注释在不兼容的浏览器中隐藏 JavaScript 代码

在与 JavaScript 不兼容的浏览器中显示 HTML 文档时，通常希望显示某些信息告诉用户他们的浏览器与您的程序不兼容。可以使用<NOSCRIPT>…</NOSCRIPT>标签对向使用不兼容浏览器的用户显示一条替换信息。<NOSCRIPT>…</NOSCRIPT>标签对通常跟在<SCRIPT>…</SCRIPT>标签对之后。图 1-20 说明了如何使用<NOSCRIPT>标签。

提示：如果用户停用了他们的浏览器的 JavaScript 支持，<NOSCIRPT>标签中替换信息也会被显示。

下面将会修改 MyFirstJavaScript.html 文件以便把它从不兼容的浏览器中隐藏起来，并使用<NOSCRIPT>标签显示一条替换信息。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!--This line starts the HTML comment block
document.writeln("Your order has been confirmed.");
document.writeln("Thank you for your business.");
This line ends the HTML comment block-->
</SCRIPT>
<NOSCRIPT>
This line is displayed if a browser does not support JavaScript<BR>
</NOSCRIPT>
```

图 1-20　使用<NOSCRIPT>标签的 JavaScript 代码

修改 MyFirstJavaScript.html 文件以便把它从不兼容的浏览器中隐藏起来，并使用<NOSCRIPT>标签显示一条替换信息：

1. 在文本编辑器或 HTML 编辑器中打开 MyFirstJavaScript.html 文件。

2. 把光标置于包含起始<SCRIPT>标签的行尾，回车，然后键入<!--开始一个把 JavaScript 从不兼容的浏览器中隐藏起来的 HTML 注释块。

3. 把光标置于 document.writeln("This is the first line in my JavaScript file.");//Line2 语句的行尾，回车，然后键入-->结束 HTML 注释块。

4. 把光标置于</PRE>标签的行尾，回车插入一个新行，然后键入如下代码，为不支持 JavaScript的浏览器显示一个消息。

```
<NOSCRIPT>
Your browser does not support JavaScript.<BR>
</NOSCRIPT>
```

5. 保存 MyFirstJavaScript.html 文件并在 Web 浏览器中打开它。如果使用的是最新版本的 Navigator 或 Internet Explorer，JavaScript 部分就会正常地执行。但是，如果使用了一个不支持 JavaScript 的浏览器，就会看到“ Your browser does not support JavaScript ”这样的消息。

1.2.5　在<HEAD>或<BODY>段放置 JavaScript

Web 浏览器按照标签在 HTML 文档出现的顺序翻译标签。如果在一个 HTML 文档中有多个 JavaScript 代码段，每一段也会按照出现的顺序执行。例如，在以下的代码中，嵌入的 JavaScript 代码会在 JavaScript 源代码文件调用之前执行，因为嵌入的代码首先出现。

```
The following embedded JavaScript code executes first. <BR>
<SCRIPT LANGUAGE="JavaScript1.2">
document.writeln("First JavaScript code section in document");
</SCRIPT> <BR>
The following JavaScript source file executes after the embedded
JavaScript code. <BR>
<SCRIPT LANGUAGE="JavaScript1.2" SRC="javascriptsource.js">
</SCRIPT>
```

浏览器执行 JavaScript 代码的顺序也依赖于 JavaScript 代码放置于 HTML 文档中的哪一部分。HTML 文档通常包括一个<HEAD>段和一个<BODY>段。<HEAD>段包含被浏览器使用的信息，并在<BODY>段之前翻译。<BODY>段通常包含用于显示的 Web 页面的内容。JavaScript 可以置于这两段中或者两段之间。在哪儿放置 JavaScript 代码是变化的，要依赖于您所写的程序。

因为 HTML 文档的<HEAD>段在<BODY>段之前翻译，所以应该尽可能多的把 JavaScript 代码放在<HEAD>段中。当把 JavaScript 代码放在<HEAD>段中时，它会在被显示的 HTML 文档主体之前处理。如果需要为<BODY>段的 JavaScript 代码完成所需的后台任务，可能会把 JavaScript 代码置于<HEAD>段中。

1.2.6 总结

- ✧ HTML 文档中组成 JavaScript 程序的语句包含在<SCRIPT>…</SCRIPT>标签对中。
- ✧ <SCRIPT>标签的 LANGUAGE 属性告诉浏览器使用的是哪种脚本语言和脚本语言的版本。
- ✧ JavaScript 是大多数 Web 浏览器的默认脚本语言。
- ✧ 如果使用的 Web 浏览器不支持所指明的 JavaScript 版本，浏览器就会忽略<SCRIPT>...</SCRIPT>标签对中的所有语句。
- ✧ 如果希望 Web 页能与老版本的 Web 浏览器兼容，应该指明老版本浏览器支持的 JavaScript 版本。
- ✧ Document 对象代表浏览器窗口中的内容。
- ✧ 可以使用 Document 对象的 write()方法和 writeln()方法在 Web 页添加新的文本。
- ✧ 参数就是能够传递给一个方法的任何类型的信息。
- ✧ 文本字符串或文字串，就是包含在双引号之间的文本。
- ✧ write()和 writeln()方法实现的功能和在 HTML 文档中手工添加文本基本相同。write()和 writeln()方法之间惟一的区别就是 writeln()方法在文本行的结尾加一个回车换行符。
- ✧ 不像 HTML，JavaScript 是对大小写敏感的。

- ✧ 也可以把 JavaScript 代码保存在一个被称为源代码文件的外部文件里。JavaScript 源代码文件通常使用.js 扩展文件名命名，其中仅包含有 JavaScript 语句，它不包含 HTML 中的<SCRIPT>…</SCRIPT>标签对。
- ✧ 访问保存在外部文件里的 JavaScript 代码需要使用<SCRIPT>标签的 SRC 属性。SRC 属性接受一个指明 JavaScript 源代码文件 URL 或目录位置的文本字符串。
- ✧ JavaScript 源代码文件中不能包含 HTML 标签。
- ✧ HTML 文档中同时使用嵌入式 JavaScript 代码和 JavaScript 源代码文件。
- ✧ 在 HTML 文档中包含多个 JavaScript 代码段时，每一段都必须使用<SCRIPT>…</SCRIPT>标签对。
- ✧ 注释就是放在代码中的非打印行，其中包含各种标注。
- ✧ 行注释是通过在想要用作注释的文本前加上两个斜杠//来创建。
- ✧ 块注释可以跨越多行，通过在块中第一行前添加/*创建。在块中最后的文本之后键入*/结束注释块。
- ✧ 为了在不兼容的浏览器中隐藏嵌入的 JavaScript 代码，就需要把<SCRIPT>…</SCRIPT>标签对置于一个 HTML 注释块（<!--…-->）之中。
- ✧ 大多数 Web 浏览器不会显示用 HTML 注释标签隔开的行。但是，与 JavaScript 兼容的浏览器会忽略 HTML 注释标签，正常地执行 JavaScript 代码。
- ✧ 可以使用<NOSCRIPT>…</NOSCRIPT>标签对向使用不兼容浏览器的用户显示一条替换信息。<NOSCRIPT>…</NOSCRIPT>标签对通常跟在<SCRIPT>…</SCRIPT>标签对之后。
- ✧ 如果在一个 HTML 文档中有多个 JavaScript 代码段，每一段也会按照出现的顺序执行。
- ✧ 因为 HTML 文档的<HEAD>段在<BODY>段之前翻译，所以应该尽可能多地把 JavaScript 代码放在<HEAD>段中。

1.2.7　问题

1. 一个 HTML 文档中的脚本代码位于_____。

a. 在<SCRIPT>标签的结束括号内
b. 在<SCRIPT>…</SCRIPT>标签对之间
c. 在起始<SCRIPT>标签之前
d. 在结束<SCRIPT>标签之后

2. <SCRIPT>标签_____。

a. 只用于 JavaScript
b. 只用于 VBScript
c. 可以用于 JavaScript 和 VBScript
d. 既不用于 JavaScript，也不用于 VBScript

3. 以下哪个选项对<SCRIPT>标签的 LANGUAGE 属性是无效的？

a. JavaScript1.2

b. JAVASCRIPT1.2

c. javascript1.2

d. JavaScript 1.2

4. 如果一个 Web 浏览器不支持 LANGUAGE 属性指定的 JavaScript 版本，那么包含在<SCRIPT>…</SCRIPT>标签对之间的 JavaScript 语句_____。

a. 被转换为所支持的版本

b. 关闭 Web 浏览器

c. 被忽略

d. 被显示为文本

5. 一个_____是指可以被当作一个独立的单元或组件的程序代码和数据。

a. 图标

b. 过程

c. 封装单元

d. 对象

6. 代表浏览器窗口的 JavaScript 对象叫做_____对象。

a. Document

b. HTML

c. Browser

d. Contents

7. 在 JavaScript 中，可以使用 write()方法或_____方法在 Web 页面里创建新的文本。

a. output()

b. writeln()

c. print()

d. println()

8. 以下哪个选项使用的是在文字串中包含引用文本的正确语法？

a. "this is a ""quoted"" string"

b. 'this is a 'quoted' string'

c. "this is a "quoted" string"

d. "this is a 'quoted' string"

9. _____标签对通知 Web 浏览器包含在其中的所有文本和断行都按照原样提交。

a. <FORMAT>…</FORMAT>

b. <CONTAINER>…</CONTAINER>

c. <COMPOSE>…</COMPOSE>

d. <PRE>…</PRE>

10. 下面那个语句是正确的？

a. DOCUMENT.write("Hello World")

b. Document.Write("Hello World")

c. document.write("Hello World")

d. document.WRITE("Hello World")

11. JavaScript 源代码文件使用<SCRIPT>标签的_____ 属性调用。

a. LANGUAGE

b. FILE

c. SOURCE

d. SRC

12. JavaScript 源代码文件_____。

a. 能够包含 HTML 标签

b. 不能包含 HTML 标签

c. 能够包含某些 HTML 标签

d. 不能包含 JavaScript 语句

13. 一个 HTML 文档调用 JavaScript 源代码文件时，<SCRIPT>…</SCRIPT>标签对_____。

a. 位于 HTML 文档之中

b. 位于 JavaScript 源代码文件之中

c. 位于 HTML 文档和 JavaScript 源代码文件之中

d. 不是必须的

14. 哪种情况下不会使用 JavaScript 源代码文件？

a. 在不兼容的浏览器中使用 JavaScript 代码时

b. 当 JavaScript 源代码文件被多个 HTML 文档共享时

c. 当 JavaScript 代码较短而且不共享时

d. 当希望与其他程序员共享代码时

15. HTML 文档可以包含_____。

a. 嵌入的 JavaScript 代码，但不能包含 JavaScript 源代码文件

b. JavaScript 源代码文件，但不能包含嵌入的 JavaScript 代码

c. JavaScript 代码或者 JavaScript 源代码文件

d. 嵌入的 JavaScript 代码和 JavaScript 源代码文件

16. 在 JavaScript 中可以在想要当作注释的文本之前添加_____创建行注释。

a. ||

b. **

c. //

d. \\

17. 块注释以/*开始，以_____结束。

a. */

b. /*

c. //

d. **

18. 使用_____可以把 JavaScript 代码从不兼容的浏览器中隐藏起来。

a. HTML 过滤器

b. <MASK>标签

c. JavaScript 注释标签

d. HTML 注释标签

19. 为不兼容浏览器用户显示替换信息使用_____标签对。

a. <NOSCRIPT>…</NOSCRIPT>

b. <SUBMESSAGE>…</SUBMESSAGE>

c. <MESSAGEBOX>…</MESSAGEBOX>

d. <NOJAVASCRIPT>…</NOJAVASCRIPT>

20. HTML 文档中的 JavaScript 代码段如何执行？

a. 所有嵌入 JavaScript 代码首先执行

b. 所有 JavaScript 源代码文件首先执行

c. 每一个 JavaScript 代码段按照把他们添加到 HTML 文档中的顺序执行

d. 每一个 JavaScript 代码段按照出现的顺序执行

21. 在一个 HTML 文档中，JavaScript_____。

a. 不能被放在<HEAD>或<BODY>段

b. 能够放在<HEAD>或<BODY>段

c. 只能放在<HEAD>段

d. 只能放在<BODY>段

1.2.8 练习

1. 说明什么时候应该使用嵌入 JavaScript，什么时候应该使用 JavaScript 源代码文件。

2. 按照以下说明创建一个 HTML 文档。使用一个 HTML 标签创建文档标题。使用嵌入 JavaScript 代码创建第一个文本行，使用 JavaScript 源代码文件创建第二个文本行。确保在<SCRIPT>标签中包含 LANGUAGE 属性。包含一个带有<TITLE>的<HEAD>段；使用“Tutorial1Exercise2”作为标题。添加一个说明姓名和日期的注释。使用 JavaScript 注释在嵌入的 JavaScript 代码和 JavaScript 源代码文件中添加相同的注释信息。使用 HTML 注释

标签把 JavaScript 代码在不兼容的浏览器中隐藏起来，并创建一个<NOSCRIPT>段，说明为“Your browser does not support JavaScript.”。把文件保存为盘上 Tutorial.01 目录下的 Tut1Ex2.html 文件。图 1-21 显示了文档应该显示出来的样子。

图 1-21　练习 2

3. 创建一个打印各大洲名称的 HTML 文档。使用一个 HTML 标签创建文档标题。使用标准 HTML 标签在每一行创建编号，但使用嵌入 JavaScript 代码创建洲名（可能需要使用多个 JavaScript 段）。确保在<SCRIPT>标签中包含 LANGUAGE 属性。包含一个带有<TITLE>的<HEAD>段；使用“Tutorial1Exercise3”作为标题。添加一个说明姓名和日期的注释。使用 JavaScript 注释在嵌入的 JavaScript 代码和 JavaScript 源代码文件中添加相同的注释信息。使用 HTML 注释标签把 JavaScript 代码在不兼容的浏览器中隐藏起来，并创建一个<NOSCRIPT>段，说明为“Your browser does not support JavaScript.”。把文件保存为盘上 Tutorial.01 目录下的 Tut1Ex3.html 文件。HTML 文档看起来应该和图 1-22 相似。

图 1-22　练习 3

4. 创建一个打印宪法前言的 HTML 文档。同时使用 HTML 标签和 JavaScript 代码创建段落，并对它进行格式化。确保在<SCRIPT>标签中包含 LANGUAGE 属性。包含一个带有<TITLE>的<HEAD>段；使用“Tutorial1Exercise4”作为标题。添加一个说明姓名和日期的注释。使用 JavaScript 注释在嵌入的 JavaScript 代码和 JavaScript 源代码文件中添加相同的注释信息。使用 HTML 注释标签把 JavaScript 代码在不兼容的浏览器中隐藏起来，并创建一个<NOSCRIPT>段，说明为“Your browser does not support JavaScript.”。把文件保存为盘上 Tutorial.01 目录下的 Tut1Ex4.html 文件。HTML 文档看起来应该类似于图 1-23。

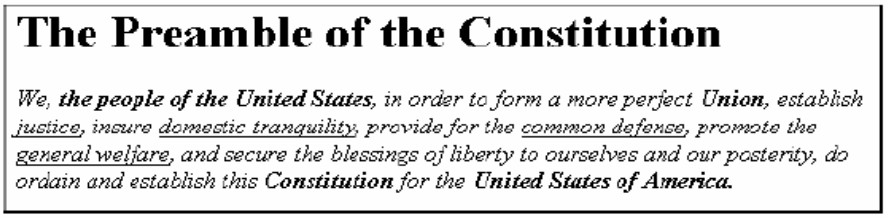

The Preamble of the Constitution
We, the people of the United States, in order to form a more perfect Union, establish justice, insure domestic tranquility, provide for the common defense, promote the general welfare, and secure the blessings of liberty to ourselves and our posterity, do ordain and establish this Constitution for the United States of America.

图 1-23　练习 4

第 2 章　变量、函数、对象和事件

案例

图像映射是 Web 站点常用的功能。它由一幅被分为多个区域的图片组成，而且每个区域都与一个 URL 相关。单击每一个区域都可以打开与它相关的 URL。使用 JavaScript 代码可以在用户的鼠标移过或者离开图像映射的一个区域时完成不同的任务。

一个汽车租赁代理商——WebAdventure 的一个客户，想把他们企业主页做成一个图像映射。他们希望能够让顾客知道他们可以使用 Web 站点在北美找到最近的获取和交还汽车的地点。您的任务是创建一个图像映射显示国家的名称，并在鼠标经过时改变该国家的颜色。

预览 NorthAmericaImageMap.html 文件

在本章中，将创建一个名为 NorthAmericaImageMap.html 的 HTML 文档，该文档使用 JavaScript 显示北美各国家的名称，并在鼠标经过时改变所在国家的颜色。

1. 在 Web 浏览器中打开盘上 Tutorial.02 目录中的 NorthAmericaImageMap.html 文件。它会显示一幅北美的图片。图 2-1 显示了这个程序在 Web 浏览器中的实况。

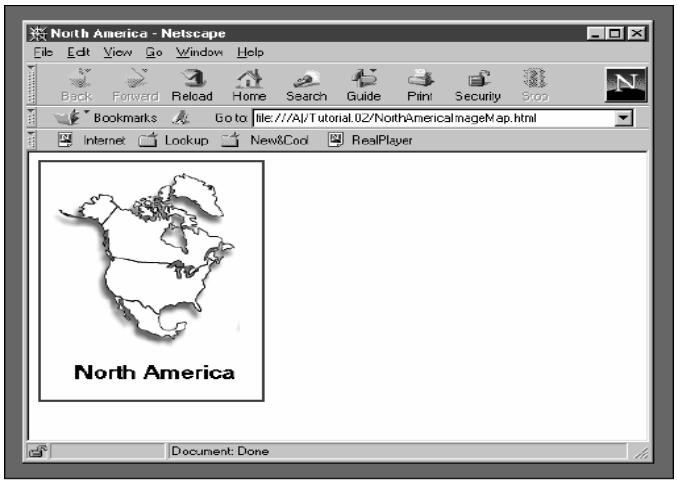


图 2-1　NorthAmericaImageMap.html

2. 移动鼠标经过地图上的每一个区域。当鼠标进入一个国家时，该国家被高亮度显示，而且图片底部的文本改为该国家的名称。当把鼠标移到高亮度区域之外时，会再次显示最初的北美图片。

3. 完成之后，关闭浏览器窗口。

4. 接下来，在文本编辑器或 HTML 编辑器中打开 NorthAmericaImageMap.html 文件，分析一下代码。注意以“ function ”开始的语句。函数中包含了在图像映射中改变高亮度区域的代码。图像映射自身是在<BODY>段中使用<IMAGE>、<MAP>和<AREA>标签创建的。

5. 分析完代码之后关闭文本编辑器或 HTML 编辑器。

2.1 使用变量、函数和对象进行工作

本节目标

在本节会学到：

- ✧ 如何声明和使用变量
- ✧ 如何定义函数
- ✧ 如何调用函数
- ✧ 如何使用 JavaScript 对象
- ✧ 如何使用对象继承和对象原型
- ✧ 如何使用对象方法
- ✧ 关于变量作用域

2.1.1 变量

程序设计最重要的特征之一就是存储和操作计算机内存单元中值的能力。存储在计算机内存单元中的值被称为变量。一个特定变量中的数据是经常改变的。例如，可能在程序中创建了一个存储当前时间的变量。每次程序运行的时候，时间都是不同的，所以变量的值会改变。在另外一个例子中，工资单程序把雇员姓名赋给变量 ExployeeName。变量 ExployeeName 引用的内存单元在不同的时刻可能包含不同的值（公司每一个雇员都有一个不同的值）。

在 JavaScript 中，使用保留字 var 创建变量。保留字或关键字是 JavaScript 语法的组成部分。保留字不能用作变量名。图 2-2 中列出了 JavaScript 的保留字。

提示：图 2-2 中的某些保留字现在并没有使用，它们是为将来的使用保留的。

使用保留字 var 创建一个变量，可以在变量声明时使用语法 var variable_name=value;

把值赋给一个变量。变量声明中的等号把值赋给变量。这种用法与核算公式中等号的标准用法并不相同。

abstract	char	do	finally
boolean	class	double	float
break	const	else	for
byte	continue	extends	function
case	default	false	goto
catch	delete	final	if
implements	new	static	true
import	null	super	try
in	package	switch	typeof
instanceof	private	synchronized	var
int	protected	this	while
interface	public	throw	with
long	return	throws	
native	short	transient	

图 2-2　JavaScript 保留字

赋给变量的值可以是文字串或数值。例如，语句 var myVariable = "Hello";把文字串 Hello 赋给变量 myVariable。语句 var myVariable = 100;把数值 100 赋给变量 myVariable。

提示：声明变量时 var 关键字并不是必须的。然而，忽略 var 关键字将影响变量在程序中使用的位置。不管是在程序的何处使用变量，使用 var 关键字声明变量都是一种好的编程习惯。

在同一条语句中可以使用一个 var 关键字声明多个变量，由变量名和赋给它们的值组成的赋值语句用逗号隔开。例如，下面的语句使用一个 var 关键字创建多个变量。

```
var firstVar = "text"，secondVar = 100，thirdVar = 2.5;
```

注意前面的例子中每一个变量都被赋给了一个值。尽管可以在声明变量的时候为它赋值，但这不是必须的。程序可能会在稍后为它赋值，也可能使用一个变量存储用户的输入。声明一个没有赋值的变量必须使用 var 关键字。

提示：声明一个没有赋值的变量时，变量是未定义的。当试图使用一个没有声明的变量或者使用一个不存在的变量或属性时，就会返回一个未定义的值。

不管声明一个变量时是否为它赋了值，都可以在程序中的任何地方使用赋值语句改变变量值，赋值语句由变量名、等号以及赋给变量的值组成。下面的代码声明了一个名为 myDog 的变量，并为它分配了一个初始值“ Golden Retriever ”，然后使用 document.writeln() 函数打印该变量。第三条语句把 myDog 变量的值改为“ Irish Setter ”，第四条语句打印该变量的新值。myDog 变量仅用 var 关键字声明了一次。

```
var myDog ="Golden Retriever";
```

```
document.writeln(myDog);
mydog ="Irish Setter";
document.writeln(myDog);
```

分配给变量的名字是一个标识符，标识符必须由大写或小写 ASCII 字母、美元号（$）或下划线（_）开始。可以在标识符中使用数字，但不能用作第一个字符。

提示：JavaScript 不允许使用数字作为标识符的第一个字符，以便可以轻松地区分标识符和字面量。

变量的命名有一些必须遵守的规则和约定。保留字不能用作变量名，在变量名中不能使用空格。通常在变量名中使用下划线（_）来分隔独立的单词，例如 my_variable_name。另外一个惯例是变量名中的一个单词的第一个字母使用小写字母，变量名中其他单词都以大写字母开头，例如 myVariableName。图 2-3 中列出了一些合法变量名的示例，图 2-4 中则列出了一些不合法变量名的示例。

```
my_variable
$my_variable
_my_variable
my_variable_example
myVariableExample
```

图 2-3　合法变量名示例

```
%my_variable
1my_variable
#my_variable
@my_variable
~my_variable
+my_variable
```

图 2-4　不合法变量名示例

提示：某些版本的 Web 浏览器，例如 Navigator 2.0 和 Internet Explorer 3.02 不能识别变量名中的美元符号。如果希望 JavaScript 程序可以在老版本的 Web 浏览器上运行，应该避免在变量名中使用美元符号。

变量名和其他 JavaScript 代码一样，是对大小写敏感的。因此，变量 myVariable 和变量 myvariable、MyVariable 或 MYVARIABLE 具有不同的值。如果在运行 JavaScript 程序时收到一个出错信息，请检查在引用声明的变量时是否使用了正确的大小写。

2.1.2　定义函数

计算机程序里使用的单独的语句通常被组织成称为过程的逻辑单元。在 JavaScript 程序设计中，过程被称为函数。函数让您可以把一组相关的 JavaScript 语句当作一个独立的单元。函数和所有其他 JavaScript 代码一样，必须放在<SCRIPT>…</SCRIPT>标签对之内。在 JavaScript 程序中使用函数之前，首先必须创建或定义该函数。在 HTML 文档内构成函数的语句行叫做函数定义。定义函数的语法为：

```
function name_of_function(parameters){
statements;
}
```

函数定义包括三个部分：

- ✧ 保留字 function 后跟函数名。保留字 function 通知 JavaScript 解释器下面的代码是一个函数。和变量相同，分配给函数的名字也叫做标识符，用于变量命名的规则和约定同样适合于函数名。
- ✧ 包含在函数名之后的圆括号中的所有函数所需的参数。
- ✧ 位于花括号{}之内的函数语句。

参数放在函数名后的圆括号内。参数或变元是在函数内部使用的变量。例如，写了一个名叫 calculate_square_root()的函数用来计算 number 变量所包含的数字的平方根。函数名就应该写为 calculate_square_root(number)。函数可以包含多个用逗号分开的参数。包含三个数字型参数的函数 calculate_square_root()可以写为 calculate_square_root (number1, number2, number3)。

提示：参数并不是必需的。许多函数仅仅执行一项任务，并不需要外部数据。例如，一个用于向访问网站的用户显示相同信息的函数仅需要被执行，而不需要任何其他信息。

包含函数参数的圆括号后是一对包含函数语句的花括号。函数语句必须包含在花括号之内。图 2-5 是一个打印多个公司名称的函数实例。

注意图 2-5 中的函数是如何组织的。起始花括号和函数名称位于同一行，结束花括号位于函数语句之后，单独构成一行。花括号中的每条语句都缩进 1.5 英寸。这种结构是许多 JavaScript 程序员的首选格式。回忆一下，JavaScript 会忽略回车、空格和制表符，所以对于简单的函数来说，在同一行中包括函数名、花括号和语句有时要更容易一些。JavaScript 惟一的语法要求是使用分号在同一行分隔语句。

```
Function print_company_name(company1,company2,company3) {
document.writeln(company1);
document.writeln(company2);
document.writeln(company3);
}
```

图 2-5　打印多个公司名称的函数

2.1.3　调用函数

函数定义不会自动执行。创建一个函数仅仅是给函数命名、指明函数的参数以及组织要执行的语句。要想执行一个函数，必须在程序的其他地方唤醒或调用此函数。调用一个函数，需要创建一条包括函数名称和包含分配给函数参数的变量或值的圆括号的语句。把变量或值发送给被调用函数的参数称为参数传递，它使参数获得被传递的变量的值。

通常函数在<HEAD>段创建，在<BODY>调用。HTML 文档的<HEAD>段总是在<BODY>段之前翻译，所以把函数放在<HEAD>段，函数调用放在<BODY>段能够确保函数在实际调用之前已经创建。如果程序试图在创建一个函数之前调用它，就会产生一个错误。图 2-6 所示是一个打印公司名称的 JavaScript 程序。图 2-7 是其输出。注意，这个 HTML 文档中<HEAD>段定义的函数是在<BODY>段调用的。

```
<HTML>
<HEAD>
<TITLE>Print Company Name Function</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--HIDE FROM INCOMPATIBLE BROWSERS
function print_company_name(company_name) {
document.writeln(company_name);
}
//STOP HIDING FROM INCOMPATIBLE BROWSERS-->
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--HIDE FROM INCOMPATIBLE BROWSERS
print_company_name("My company");
//STOP HIDING FROM INCOMPATIBLE BROWSERS-->
</SCRIPT>
</BODY>
</HTML>
```

图 2-6　在<BODY>段调用的 JavaScript 函数

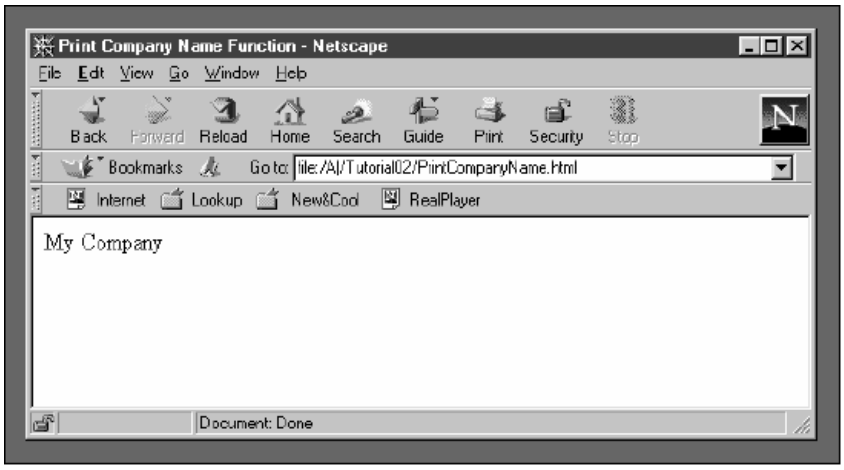


图 2-7　在<BODY>段调用的 JavaScript 函数的输出

在图 2-6 所示的程序中，函数调用语句把“My company”传给函数 print_company_name()，该函数收到字符串后，把它分配给 company_name 参数变量。

提示：一个 JavaScript 程序由 HTML 文档中所有的<SCRIPT>段组成。每一个独立的<SCRIPT>段不必是一个独立的 JavaScript 程序（如果 HTML 文档没有其他<SCRIPT>段，一个<SCRIPT>段也可以是一个独立的 JavaScript 程序）。

很多情况下，可能希望在一个函数中接收来自另外一个函数的值，并把它用在其他代码中。例如，如果有一个函数对传给它的数字做计算，可能希望收到计算结果。假设有一个函数将传给它的一系列数字计算平均值，如果看不到计算结果，这个函数是没有用的。为了在调用语句中返回一个值，需要给调用语句赋一个变量。下面的语句调用函数 average_numbers()并把返回值赋给变量 returnValue，同时这条语句把三个字面量传给函数。

```
var returnValue = average_numbers(1, 2, 3);
```

为了能够把值返回给 returnValue 变量，必须在 average_numbers()函数中包含 return 语句。下面的代码是 average_numbers()函数，它计算三个数字的平均值并使用 return 语句把 result 变量中的值返回给调用语句。

```
function average_numbers(a, b, c){
    var sum_of_numbers = a + b + c;
    var result = sum_of_numbers /3;
    return result;
}
```

提示：返回值对函数来说不是必需的。

提示：在第 3 章会学到更多关于计算的知识。

从函数中返回值的变量名和接收返回值的变量名可以相同。例如，在前面的例子中，函数中的 return 语句和 74 ,调用语句中的变量名可以都是 returnValue。而且，当把变量当作参数传给函数时，被传递的变量名和函数参数名也可以是相同的。如果想把变量而不是字面量传递给函数 average_numbers()，即使参数名本身就是 a、b、c，也可以使用语句 average_numbers(a,b,c);。但是，大多数程序员通常在他们的代码中使用惟一的名字来标识特定的变量。

提示：使用惟一的名字标识特定的变量能够更容易地理解程序员的逻辑，并有助于调试过程。

并不是所有的函数都需要返回值。例如，一个改变 HTML 文档背景色或者其他一些不会创建或返回有用值的函数并不需要返回值。如果不需要从函数中接收返回值，就不必给函数调用语句赋一个变量。例如，如果想调用 average_numbers()函数计算三个字面量 2、3、4 的平均值，但是不需要返回值，可以输入 average_numbers(2,3,4);，并不需要把它赋给 returnValue 变量。

提示：当函数执行计算平均值之类的运算时，通常希望能够得到一个返回值。

下面会创建一个包含两个函数的 JavaScript 程序。第一个函数被调用时会打印一条消息，第二个函数返回一个在调用语句之后打印的值。

创建一个包含两个函数的 JavaScript 程序：

1. 在文本编辑器或 HTML 编辑器创建一个新文档。

2. 输入起始<HTML>和<HEAD>标签，然后输入<TITLE>…<TITLE>标签对，如下所示：

```
<HTML>
<HEAD>
<TITLE>Two Function Program</TITLE>
```

3. 输入起始<SCRIPT>标签和 HTML 注释，把代码从不兼容的浏览器中隐藏起来。

4. 输入第一个函数，它使用调用语句传递给它的参数在屏幕上打印一条消息。

```
function print_message(first_message){
    document.writeln(first_message);
}
```

5. 输入显示下一条消息的第二个函数。这条语句仅仅把文字串“ This message was returned from a function ”返回给调用语句。

```
function return_message(second_message){
```

```
    return "This message was returned from a function";
}
```

6. 输入下面一行结束<SCRIPT>段。

```
//STOP HIDING FROM INCOMPATIBLE BROWSERS -- >
</SCRIPT>
```

7. 输入</HEAD>结束<HEAD>…</HEAD>标签对。

8. 输入如下代码开始 HTML 文档体，并创建一个预格式化容器。

```
<BODY>
<PRE>
```

9. 输入调用<HEAD>部分中函数的 JavaScript 段的起始语句。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

10. 输入下面的两条语句调用<HEAD>段的两个函数。第一条语句把文本字符串“ This text was printed from a function ”传递给第一个函数，而且没有返回值。第二条语句把返回值赋给变量 return_value，但是不传递任何参数。

```
print_message("This text was printed from a function");
var return_value = return_message();
```

11. 用 document.writeln(return_value);语句把 return_value 变量值输出到屏幕上。

12. 输入下面的代码结束 HTML 注释并结束<SCRIPT>标签对。

```
//STOP HIDING FROM INCOMPATIBLE BROWSERS -- >
</SCRIPT>
```

13. 结束<PRE>、<BODY>和<HTML>标签。

```
</PRE>
</BODY>
</HTML>
```

14. 把文件保存为盘上 Tutorial.02 目录下的 TwoFunctionsProgram.html 文件。在 Web 浏览器中打开 TwoFunctionsProgram.html 文件，图 2-8 显示了 TwoFunctionsProgram.html 文件。

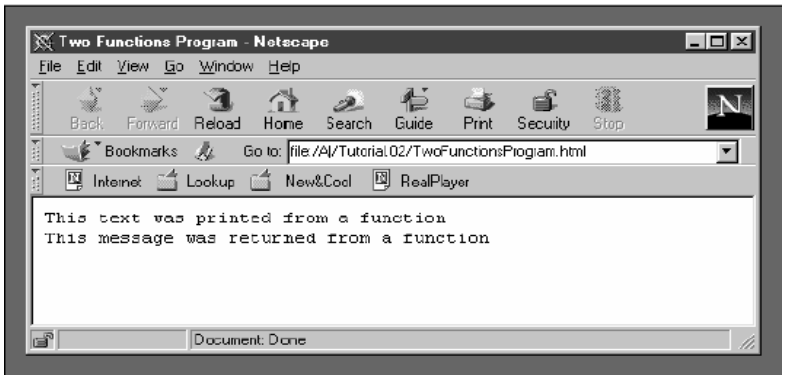


图 2-8 TwoFunctionsProgram.html

帮助：如果收到一个错误信息，请检查 JavaScript 代码大小写是否正确——记住，JavaScript 是对大小写敏感的。同时检查一下是否输入了所有的 HTML 起始和结束标签。

15. 关闭 Web 浏览器窗口和文本编辑器。

2.1.4 理解 JavaScript 对象

传统的面向对象编程语言，例如 C++和 Java，都会创建用于访问过程和数据的对象。对象是基于类的。在面向对象程序设计中，数据、过程和其他属性都包含在通常所说的类中。对象是类的一个实例，它继承了类的过程和数据，不能直接访问一个类。对象也可以基于其他对象。基于其他类或对象的对象称为后代对象。类似地，对象所继承的类或对象称为祖先类对象。

比较而言，JavaScript 对象是基于构造器函数的。作为对象基础的这个函数被称为对象定义，或者构造器函数。使用构造器函数创建一个新对象时，可以说是实例化了一个新对象或者扩展了旧对象。和传统的基于类的对象一样，JavaScript 对象继承了它们所基于的构造器函数的所有数据和过程。任何 JavaScript 函数都可以用作构造器。

提示：构造器函数更像是作为对象基础的一个模板，而不是用来实例化对象的类。

构造器函数包含两种元素：属性和方法。属性是构造器函数内部的变量。这些变量或属性可以认为是通过构造器函数创建的所有对象的数据。方法是一个在对象内部调用的函数——可以是内置 JavaScript 函数或自己创建的函数。

提示：属性也称作域。

下面的代码是一个名为 Animal 的构造器函数，它包含三个属性：animal_type、animal_sound 和 animal_transport_mode。

```
function Animal(type, sound, transport_mode) {
```

```
    this.animal_type = type; // dog, cat, etc.
    this.animal_sound = sound;       // woof, meow, etc.
    this.animal_transport_mode = transport_mode;
        // walk/run, fly, swim
}
```

提示：在传统的面向对象程序设计语言中，类名通常以大写字母开头。因为构造器函数相当于类，所以构造器函数名通常也以大写字母开头，以便区别于代码中的普通函数。

注意前面例子中的 this 关键字。this 关键字所代表的是调用构造器函数的当前对象。函数中的三条语句把三个参数 type、sound 和 transport_mode 赋给对象（this）的属性 animal_type、animal_sound 和 animal_transport_mode，该对象使用构造器函数进行实例化。this 的使用是构造器函数和标准函数最大的区别之一。因为标准函数不能作为对象的基，所以标准函数不能包含 this 引用。

提示：this 也可以用在<FORM>标签中代表包含一个对象的表单。在第 6 章会讲到有关表单的知识。

对象可以使用 new 关键字从构造器函数创建。下面的代码使用构造器函数 Animal 创建一个名为 pet 的新对象，并把三个适当的参数赋给对象的属性：

```
pet = new Animal("dog", "woof", "walk/run");
```

pet 对象现在具有三个属性：type、sound 和 transport_mode。访问对象的属性，只需在对象名后加点并加上属性名。例如，访问 pet 对象的 sound 属性使用 pet.sound。与 document.write()这样的方法不同，属性后没有圆括号。如果在属性后加上圆括号，JavaScript 就会试图使用这个名字定位一个方法（或函数）。例如，如果写的不是 pet.sound 而是 pet.sound()，JavaScript 会假设要运行 pet 对象的 sound 方法，而不是访问 pet 对象的 sound 属性。

2.1.5　对象继承和原型

对象会从实例化它们的构造器函数中继承属性和方法。如果基于 Animal 构造器函数实例化一个新的对象 cat，这个新对象中就会包含属性 animal_type、animal_sound 和 animal_transport_mode。实例化一个新对象之后，还可以使用点号为对象增加新的属性。下面的代码基于 Animal 构造器函数创建了一个名为 cat 的新对象，然后为它增加了一个新的属性 size。

```
cat = new Animal("feline", "meow", "walk/run");
cat.size = "fat";
```

构造器函数的参数并不是必须的，如 Animal 构造器函数中的属性 type、sound 和

transport_mode。实例化一个对象时也不是必须要向构造器函数传递参数。例如，可以使用语句 cat = new Animal(); 来实例化 cat 对象，稍后再为它分配属性值。但是，如果试图使用一个没有分配值的属性，就会收到一个特殊的值“ undefined ”。如果为 cat 对象增加了一个 size 属性，但是没有为它赋值，在试图使用 document.write(cat.size);语句打印 size 属性时，就会打印出“ undefined ”。

如果为一个从构造器扩展的对象增加一个新的属性，那么这个新的属性只能在这个特定的对象中使用。这个属性在构造器函数和其他任何从同一个构造器扩展的对象中都是不可用的。但是，如果使用原型属性，创建的任何新属性在构造器函数和其他任何扩展该构造器的对象中都可以使用。原型属性是一个内置属性，它指定了对象所扩展的构造器函数。下面的代码为 Animal 构造器函数添加一个新的属性 size，这个新属性是 cat 对象的原型属性。通过使用原型属性，所有扩展 Animal 构造器函数的对象就可以访问 size 属性。

```
cat = new Animal("feline", "meow", "walk/run");
cat.prototype.size = "fat";
```

在这种情况下，所有的 Animal 对象的 size 属性都是“ fat ”。因为并不是所有的动物都能用胖来形容，所以可以使用语句 cat.prototype.size= " "; 为 size 属性赋一个空值，然后为每一个对象的 size 属性单独赋值。给 cat 对象的 size 属性赋值的语句可以是 cat.size = "fat";。

对象定义可以扩展其他对象定义。以 Animal 构造器函数为例，它包含三个可以用于所有动物的通用属性。有时可能需要创建扩展 Animal 对象的附加对象定义，其中包含专门用于某几个动物的属性。为了使用附加对象定义来扩展对象定义，必须使用原型属性，并在后面跟上 new 关键字以及被扩展的对象定义名称。图 2-9 显示了两个附加对象定义 WildAnimal 和 FarmAnimal，它们都是从 Animal 扩展而来的，而且都包含每种类型动物特有的属性。

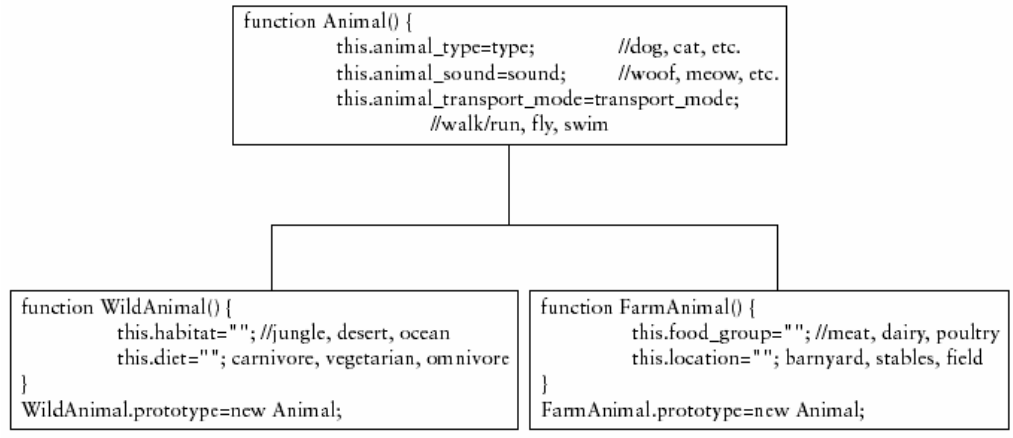


图 2-9 两个扩展其他对象定义的对象定义

使用 WildAnimal 和 FarmAnimal 实例化的对象都包含 Animal 构造器函数中的三个属

性，并且具有了每种对象特有的一些属性。例如，下面的代码使用 FarmAnimal 实例化一个对象，并为该对象的属性赋值。

```
chicken = new FarmAnimal();
// Animal object definition
chicken.animal_type = "chicken";
// Animal object definition
chicken.animal_sound = "cluck";
// Animal object definition
chicken.animal_transport_mode = "walk/fly";
// FarmAnimal object definition
chicken.location = "barnyard";
// FarmAnimal object definition
chicken.food_group = "poultry"
// FarmAnimal object definition
```

提示：某些面向对象程序设计语言允许对象继承多个对象定义。但是，JavaScript 只允许对象继承一个对象定义。

接下来会创建一个演示对象和继承的 JavaScript 程序。这个程序包含一个 Company 对象定义。Company 对象定义中包含几个可以用于公司所有部门的属性。还会扩展 Company 对象，创建两个对象定义 Sales 和 Production。Sales 和 Production 对象定义中都包含每个部门特有的属性。对象定义创建之后，会创建每一个对象的实例，并打印相关属性。

创建一个演示对象和继承的 JavaScript 程序：

1. 启动文本编辑器或 HTML 编辑器，创建一个新文档。如果文本编辑器已经打开，就创建一个新文档。

2. 输入起始<HTML>、<HEAD>和<TITLE>标签。

```
<HTML>
<HEAD>
<TITLE>Company Objects Program</TITLE>
```

3. 输入起始<SCRIPT>标签和 HTML 注释，把代码从不兼容的浏览器中隐藏起来。

4. 输入第一个构造器函数，它创建可以用于公司所有部门的属性。

```
function Company(){
   this.company_name = "WebAdventure, Inc. ";
   this.company_products = "Internet services";
}
```

5. 接下来，创建 Sales 部门的构造器函数，它扩展了 Company 对象定义。

```
function Sales(){
    this.territory = "North America";
    this.sales_reps = "50";
}
Sales.prototype = new Company();
```

6. 现在，创建 Production 部门的构造器函数，它也扩展了 Company 对象定义。

```
function Production() {
     this.facilities = "New York, Chicago, and Los Angeles";
     this.personnel = "100";
}
Production.prototype = new Company();
```

7. 输入下面代码结束<SCRIPT>和<HEAD>段。

```
//STOP HIDING FROM INCOMPATIBLE BROWSERS -- >
</SCRIPT>
</HEAD>
```

8. 输入如下代码开始 HTML 文档体，并创建一个预格式化容器。

```
<BODY>
<PRE>
```

9. 输入 JavaScript 段起始语句，调用<HEAD>段中的构造器函数。

```
<SCRIPT LANGUAGE=" JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

10. 输入如下代码实例化一个新的 Sales 对象并在屏幕上打印它的属性。注意 document.write()和 document.writeln()方法中使用加号合并字符串和属性。

提示：在第 3 章会学到更多有关 JavaScript 运算符的知识。

```
sales_object = new Sales();
document.writeln(sales_object.company_name);
document.writeln("Producer of " + sales_object.company_products);
document.write("With " + sales_object.sales_reps + " sales reps");
document.writeln(" in " + sales_object.territory);
```

11. 接下来，输入如下代码实例化一个新的 Production 对象并在屏幕上打印它的属性。

```
production_object = new Production();
document.write(production_object.company_name);
document.writeln(" has " + production_object.personnel + " production
personnel");
document.write("with facilities in " + production_object. facilities);
```

12. 输入下面的代码结束 HTML 注释并结束<SCRIPT>标签对。

```
//STOP HIDING FROM INCOMPATIBLE BROWSERS -- >
</SCRIPT>
```

13. 结束<PRE>、<BODY>和<HTML>标签。

```
</PRE>
</BODY>
</HTML>
```

14. 把文件保存为盘上 Tutorial.02 目录下的 CompanyObjects.html 文件，然后在 Web 浏览器中打开此文件，图 2-10 是 CompanyObjects.html 文件在 Web 浏览器中的样子。

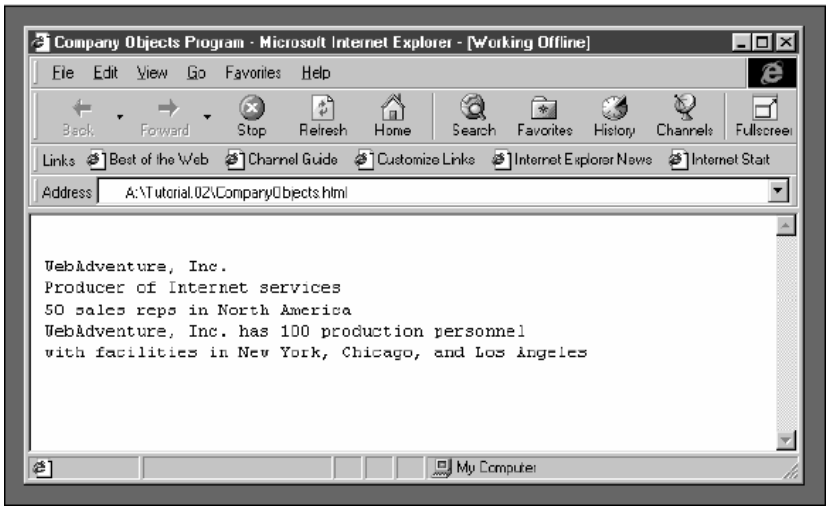


图 2-10　CompanyObjects.html

15. 关闭 Web 浏览器窗口和文本编辑器。

2.1.6　对象方法

对象方法是与一个特定对象相关的函数。创建一个用作对象方法的函数时，可以采用

和创建构造器函数相同的方式使用 this 引用。以 Animal 对象定义为例，为了创建打印它的三个属性（animal_type、animal_sound、animal_transport_mode）的方法，可以编写如下函数：

```
function displayAnimalProperties() {
   document.write(this.animal_type + "<BR>");
   document.write(this.animal_sound + "<BR>");
   document.write(this.animal_transport_mode + "<BR>");
}
```

提示：因为 displayAnimalProperties()不是一个构造器函数，所以方法名的第一个字母不是大写的。

回想一下，为了正确使用 writeln()方法，必须把它放在一个<PRE>…</PRE>标签对中。但是在很多情况下，不使用<PRE>…</PRE>要更方便一些。注意在前面的例子中并没有使用<PRE>…</PRE>，而是在每一个 document.write()方法打印的字符串中包含了一个
标签。在<SCRIPT>…</SCRIPT>中使用字面量
与使用<PRE>…</PRE>封装 document.writeln()方法获得的效果是一样的。

方法创建之后，必须使用 this.methodName = functionName;语法把它添加到构造器函数中。this 引用后的 methodName 是分配给对象内函数的名字。但是不能像调用 JavaScript 函数那样，在函数名后添加圆括号。语句 this.methodName = functionName();是错误的，因为其中包含了圆括号。为了给 Animal 函数定义添加 displayAnimalProperties()方法，需要在函数定义花括号内包含语句 this.displayAnimalProperties = displayAnimalProperties;。

基于对象定义实例化一个对象后，就可以使用对象名加点号加方法名来调用对象方法，方法名后面跟需要传给方法的所有参数的圆括号。调用对象方法的语法是 objectName.MethodName(arguments);。下面的语句基于 Animal 对象定义实例化一个新的对象，并调用该对象的 displayAnimalProperties()方法。

```
guppy = new Animal("fish", "blub", "swim");
guppy.displayAnimalProperties();
```

接下来我们会修改 CompanyObjects 程序，以便把 document.write()和 document.writeln()方法包含在 Sales 和 Production 对象自己的方法中。

修改 CompanyObjects 程序，以便把 document.write()和 document.writeln()方法包含在 Sales 和 Production 对象自己的方法中的步骤：

1. 在文本编辑器或 HTML 编辑器中打开文件 CompanyObjects.html，并立即保存为文件 CompanyObjectsWithMethods.html。

2. 在语句//STOP HIDING FROM INCOMPATIABLE BROWSERS -->前添加如下函数，该语句位于<HEAD>…</HEAD>标签对之间的<SCRIPT>段，这个函数将用作 Sales 对象的一个方法。需要注意的是该函数使用
标签来加入回车符，这是因为<SCRIPT>段

并没有位于<PRE>…</PRE>标签对之间。

```
function displaySalesInfo() {
    document.write(this.company_name + "<BR>");
    document.write("Producer of " + this.company_products + "<BR>");
    document.write("With " + this.sales_reps + "sales reps");
    document.write(" in " + this.territory + "<BR>");
}
```

3. 在 displaySalesInfo()函数后，紧接着输入以下的 displayProductionInfo()函数。

4. 在 Sales 函数定义的结束花括号之前加入 this.displaySalesInfo = displaySalesInfo;语句。

5. 在 Production 函数定义的结束花括号之前加入 this.displayProductionInfo = displayProductionInfo;语句。

6. 删除位于 <BODY>…</BODY> 中 <SCRIPT> 段的七条 document.write() 和 document.writeln() 语句。现在 <SCRIPT> 段应该仅包含两行实例化 sales_object 和 production_object 的代码。

7. 在实例化 sales_object 的语句后插入一行，并添加语句 sales_object.displaySalesInfo();。

8. 在实例化 production_object 的语句后插入一行，并添加语句 production_object.displaySalesInfo();。

9. 保存 CompanyObjectsWithMethods.html 文件，然后在 Web 浏览器中打开它。此文件看起来应该和图 2-10 相同。

10. 关闭 Web 浏览器窗口。

2.1.7　变量作用域

当在一个 JavaScript 程序，尤其是一个复杂的 JavaScript 程序中使用变量时，要特别注意变量的作用域。变量作用域是指一个声明过的变量可以用在程序的什么地方。变量的作用域可以是全局或局部的。全局变量在函数外声明，而且可以在程序的任何部分使用。局部变量在函数内部声明，而且只能在声明它的函数内部使用。当函数结束时，局部变量就不存在了。如果在声明局部变量的函数外部使用该局部变量，就会收到一条错误信息。

提示：函数声明圆括号中的参数也被当作是局部变量。

声明全局变量时，var 关键字是可选的。例如，可以把语句 var myVariable = "This is a variable";写为 myVariable = "This is a variable";。但是，在变量声明语句中使用 var 关键字总是一种好的编程习惯。

下面的代码中包含一个全局变量和含有一个局部变量的函数。全局变量和函数都位于

<HEAD>段的<SCRIPT>…</SCRIPT>标签对内。在<BODY>段调用函数时，可以成功地在函数内部打印全局变量和局部变量。结束函数调用后，全局变量也可以成功地在<BODY>段再次打印。但是，当程序试图在<BODY>段打印局部变量时，就会产生一条错误信息，这是因为在函数结束后局部变量已经不存在了。

```
<HTML>
<HEAD>
<TITLE>Variable Scope</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--HIDE FROM INCOMPATIBLE BROWSERS
var firstGlobalVariable = "First global variable";
function scopeExample() {
        secondGlobalVariable = "Second global variable";
        var localVariable = "Local variable";
        document.writeln(firstGlobalVariable);          // prints successfully
        document.writeln(secondGlobalVariale);          // prints successfully
        document.writeln(localVariable);                // prints successfully
}
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
<BODY>
<PRE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--HIDE FROM INCOMPATIBLE BROWSERS
scopeExample();
document.writeln(firstGlobalVariable);          // prints successfully
document.writeln(secondGlobalVariable);         // prints successfully
document.writeln(localVariable);                // error message
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

程序包含同名的全局变量和局部变量时，在调用函数的时候，局部变量优先于全局变量。如果函数为局部变量赋了一个新值，这个值也会成为全局变量的值。请看下面的代码，在拥有同名局部变量的函数被调用之前，全局变量 showDog 的值被赋为 Golden Retriever。一旦函数被调用，局部 showDog 变量就会被赋为 Irish Setter。函数运行结束之后，全局 showDog 变量的值也变为 Irish Setter。

```
var showDog = "Golden Retriever";
function duplicateVariableNames() {
        var showDog = "Irish Setter";
}
duplicateVariableNames();
document.writeln(showDog);          // value printed is Irish Setter
```

2.1.8　总结

- ✧ 存储在计算机内存单元中的值称为变量。
- ✧ 使用保留字 var 声明变量。
- ✧ 保留字或关键字是 JavaScript 语法的组成部分。在 JavaScript 程序中，保留字不能用作变量名和函数名。
- ✧ 函数可以把一组相关的 JavaScript 语句当作一个独立的单元。函数和所有其他 JavaScript 代码一样，必须放在<SCRIPT>…</SCRIPT>标签对之内。
- ✧ 在 JavaScript 程序中使用函数之前，首先必须创建或定义该函数。构成函数的语句称为函数定义。
- ✧ 参数或变元是在函数内部使用的一个变量。参数放在函数名后的圆括号内。
- ✧ 把变量或值发送给被调用函数的参数称为参数传递。
- ✧ 为了在调用语句中返回一个值，被调用函数中必须包含 return 语句。
- ✧ 作为对象基础的函数被称为对象定义，或构造器函数。
- ✧ 构造器函数包含两种元素：属性和方法。属性是构造器函数内部的变量。这些变量或属性可以认为是通过构造器函数创建的所有对象的数据。方法是一个在对象内部调用的函数——可以是内置 JavaScript 函数或自己创建的函数。
- ✧ this 关键字所代表的是调用构造器函数的当前对象。因为标准函数不能作为对象的基，所以标准函数不能包含 this 引用。
- ✧ 原型属性是一个内置属性，它指定了对象所扩展的构造器函数。
- ✧ 对象定义可以扩展其他对象定义。
- ✧ 对象方法本质上是与一个特定对象相关的函数。
- ✧ 调用对象方法的语法是 objectName. methodName(arguments);。
- ✧ 变量作用域是指一个声明过的变量可以用在程序的什么地方。变量的作用域可以是全局或局部的。全局变量在函数外声明，而且可以在程序的任何部分使用。局部变量在函数内部声明，而且只能在声明它的函数内部使用。

2.1.9　问题

1. 声明一个变量并赋给它一个字符串，下面哪个选项使用的语法是正确的？

a. var myVariable = "Hello";
b. var myVariable = Hello;
c. "Hello" = var myVariable;
d. var "Hello" = myVariable;

2. 下面哪一个选项是合法的变量名？

a. %variable_name
b. 1variable_name
c. variable_name
d. +variable_name

3. JavaScript 中的标识符不能以_____开头。

a. 大写或小写 ASCII 字符
b. 美元符（$）
c. 下划线（_）
d. 数字

4. _____可以把一组相关的 JavaScript 语句当作一个独立的单元。

a. 语句
b. 变量
c. 函数
d. 事件

5. HTML 文档中组成一个函数的语句被称为函数_____。

a. 段
b. 单元
c. 容器
d. 定义

6. 下面哪个选项不是函数的一部分？

a. 后跟函数名的保留字 function
b. <SCRIPT>…</SCRIPT>标签对
c. 函数所需的所有参数，包含在函数名后的圆括号中
d. 封装在花括号{}之间的函数语句

7. JavaScript 保留字可以被用作_____。

a. 函数名
b. 变量
c. a,b 都可以
d. a,b 都不可以

8. 包含在函数的圆括号中的变量称为参数或_____。

a. 域
b. 例程

c. 方法

d. 变元

9. 函数语句位于什么字符之间。

a. { }

b. [ ]

c. < >

d. ( )

10. 发送参数给被调用函数称为参数_____。

a. 产生

b. 传递

c. 发送

d. 提交

11. 为什么 JavaScript 函数应该放在 HTML 文档的<HEAD>段？

a. 不允许在<BODY>段创建函数

b. 这样做可以确保函数在调用之前创建

c. 它们不像其他 JavaScript 代码那样频繁的使用

d. 这样就不会忘记创建每一个函数

12. 作为一个对象基础的函数称为对象定义或_____。

a. 方法

b. 类

c. 构造器函数

d. 对象变量

13. 构造器中的变量是所有使用构造器函数创建的对象的数据，它称为_____。

a. 属性

b. 方法

c. 对象文件

d. 位

14. 无论是内置 JavaScript 函数还是自己创建的函数，可以在函数内部调用的函数都称为_____。

a. 过程

b. 方法

c. 构造器

d. 语句

15. this 关键字指的是_____。

a. HTML 文档

b. Web 浏览器窗口

c. 目前执行的 JavaScript 语句

d. 目前调用构造器函数的对象

16. 使用包含两个参数 color 和 engine 的构造器函数 Chevrolet 创建 my_car 对象的正确语法是：

a. my_car() = new Chevrolet "red", "V8";

b. my_car = new Chevrolet("red", "V8");

c. new Chevrolet("red", "V8") = my_car;

d. my_car("red", "V8") = new Chevrolet;

17. 使用包含两个参数的构造器函数创建对象时，应该怎么做：

a. 必须把两个参数都传递给构造器函数

b. 必须至少把一个参数传递给构造器函数

c. 把参数传递给构造器函数不是必需的

d. 在创建对象之前把参数传递给构造器函数

18. 指定一个对象所扩展的构造器的内置属性称为_____属性。

a. 起始

b. 默认

c. 源

d. 原型

19. 为对象原型 company 添加一个新属性 sales 的正确语法是：

a. company.prototype.sales = "";

b. prototype.company.sales = "";

c. sales.company.prototype = "";

d. company.prototype = sales("");

20. 为构造函数 myConstructorFunction 添加一个 myMethod 方法的正确语法是：

a. myConstructorFunction = new myMethod();

b. myMethod = this.myMethod;

c. this.myMethod = myMethod();

d. this.myMethod = myMethod;

21. 在函数外声明的变量称为_____变量。

a. 局部

b. 类

c. 程序

d. 全局

22. 局部变量必须_____声明。

a. 在函数之前

b. 使用 var 关键字

c. 在函数定义花括号内

d. 使用 local 关键字

2.1.10　练习

1. 创建一个名为 PersonalInfo.html 的 HTML 文档，该文档的<HEAD>段包含一个名为 printPersonalInfo 的函数。在 printPersonalInfo 函数中，使用 document.write()和 document.writeln()方法把您的姓名、地址、生日和社会安全号码打印到屏幕上，并在文档的<BODY>段调用这个函数。

2. 创建一个名为 FavoriteFoods.html 的 HTML 文档。在<HEAD>段创建四个函数，打印您最喜欢的食物名称。例如，编写一个包含语句 document.writeln("chinese");的函数 chinese。在<BODY>段以您最喜欢的食物开始调用每一个函数。组合字符串和 document.write()和 document.writeln()，以描述您所喜欢的食物的顺序。例如，打印“ My Favorite Food is Chinese ”字符串的 JavaScript 语句是 document.write("My Favorite Food is " + Chinese());。

3. 创建一个名为 CarObject.html 的 HTML 文档，该文档的<HEAD>段包含名为 Automobile 的构造器函数。Automobile 对象定义中包含四个属性：make、model、color 和 engine。在 HTML 文档的<BODY>段实例化一个新的 Automobile 对象，然后把自己汽车的属性赋给 Automobile 的每一个属性，并把所有属性打印在屏幕上。

4. 创建一个名为 CompanyInfo.html 的 HTML 文档，该文档的<HEAD>段包含名为 Company 的构造器函数。Company 对象定义中包含四个属性：name、products、motto 和 employees。然后创建一个打印雇员数目的方法 Employee()。在 HTML 文档的<BODY>段实例化一个新的 Company 对象，然后为 Company 的每一个属性赋值，并使用 writeln()方法把属性 name、products、motto 和 employees 打印在屏幕上。然后使用 Employee()方法打印雇员数目。最后，使用一个描述性的字符串合并打印出的属性值。例如，打印公司名称的时候，可以使用"The Company name is MyCompany."。

5. 创建一个 HTML 文档打印您最喜欢的电影名称。这个文档应该包含两个函数：一个打印您最喜欢的三个喜剧，另一个打印您最喜欢的三个戏剧。第一个函数名为 comedy，另外一个函数名为 drama。在 comedy 和 drama 函数内部把每一个电影名称赋给一个局部变量，然后打印每个局部变量。同时创建一个全局变量保存您最喜欢的电影名称。在 HTML 文档<BODY>段<SCRIPT>…</SCRIPT>标签对内调用 comedy 和 drama 函数，在两个函数调用之后打印您最喜欢的电影名称。最后把文件存为 FavoriteMovies.html。

2.2　使 用 事 件

本节目标

在本节会学到：

- ✧ 关于事件
- ✧ 关于 HTML 标签和事件
- ✧ 如何使用事件处理器
- ✧ 关于链接
- ✧ 如何使用链接事件
- ✧ 如何创建图像映射

2.2.1 理解事件

JavaScript 使 HTML 具有动态特性最重要的途径就是使用事件。使用 JavaScript 事件可以让用户和 Web 页进行交互。事件是由 JavaScript 监视的特定情况。最常见的事件就是用户所采取的行动，例如用户单击按钮时就会产生 Click 事件。可以把事件想象为一个触发器，它在特定的情况发生时执行指定的 JavaScript 代码。

事件最常见的用法之一是在用户的请求下执行某段代码。例如，目前在 Web 上可以找到一些 JavaScript 编写的计算器程序，如抵押计算器。用户输入了抵押所需的信息之后，比如利率、年数和贷款数目等，就可以单击计算器按钮计算每月抵押付款额。每月抵押付款额的计算就是由用户单击计算器按钮时所触发的事件来完成的。本章中所创建的图像映射中使用的是另外一种称为 MouseOver 的事件，它在用户移动鼠标经过图像时触发，对鼠标所经过的图像部分进行高亮度显示。

但是，JavaScript 监视的不仅仅是由用户触发的事件，JavaScript 也监视像 Load 事件这样不直接由用户触发的事件。Load 事件是在 Web 浏览器结束 HTML 文档装载时由 Web 浏览器自动触发的。Load 事件通常用来实现像动画这样的可视化效果，而且 Load 事件不应该在 Web 页面完全装载之前执行。以动画为例，实现动画效果的 JavaScript 程序只有在 Web 浏览器表单中的 Load 事件被触发之后才开始执行。表 2-1 显示了 JavaScript 事件列表以及触发条件。

表 2-1 JavaScript 事件

事　　件	触 发 时 间
abort	图像装载中断
blur	单选按钮之类的元素失去焦点
click	元素被单击
change	元素的值被改变
error	文档或图像装载时发生错误
focus	元素获得焦点
load	装载文档或图像
mouseout	鼠标从一个元素移开
mouseover	鼠标移过一个元素

续表

事　　件	触 发 时 间
reset	表单重置
select	用户选择表单中的一个域
submit	用户提交表单
unload	卸载文档

2.2.2　HTML 标签和事件

允许用户产生事件最常用的 HTML 标签之一是<INPUT>标签。<INPUT>标签创建可以与用户交互的输入域。<INPUT>标签有很多属性，其中包括 TYPE 属性。<INPUT>标签最基本的语法就是<INPUT TYPE="输入域类型">。TYPE 属性是<INPUT>标签不可少的一个栏目，而且它决定着<INPUT>标签产生的输入域的类型。例如，语句<INPUT TYPE="radio">创建一个单选按钮，<INPUT TYPE="text">创建一个文本框。<INPUT>标签放在<FORM>…</FORM>之间，它是表单中最常用的一个标签。在本书中从头到尾都会用到<INPUT>标签。

提示：在第 6 章会学到有关表单的知识以及如何在表单中使用<INPUT>标签。

下面的代码是一个包含 onBlur 事件处理器的<INPUT>标签的例子，<INPUT>标签放在<FORM>…</FORM>标签之间。Blur 事件在控制失去焦点时触发，在本例中，onBlur 事件处理器在文本框失去焦点时显示<INPUT>标签的值。

```
<FORM NAME="myForm">
    <INPUT TYPE="text" VALUE="default text"
        NAME="textButton" onBlur="alert(this.value);">
</FORM>
```

注意本例中的<FORM>和<INPUT>标签都使用了 NAME 属性，NAME 属性可以为 HTML 标签分配一个惟一的名字，以便可以在 JavaScript 代码中使用这个名字引用 HTML 标签。为了在函数中或者从另外一个标签来引用一个 HTML 标签，需要把 HTML 标签的名字附上以 Document 对象开始的任一祖先对象。这样就可以检索标签的信息或修改标签的属性，例如，在 JavaScript 代码函数中使用语句 document.myForm.textButton.value="new value";可以改变 textButton 的值为 new value。

提示：和大多数 HTML 代码不同，NAME 属性是大小写敏感的。

表 2-2 列出了各种 HTML 标签及其相关事件。

表 2-2 HTML 标签和相关事件

标　签	描　述	事　件
<A>…</A>	链接	click mouseover mouseout
<IMG>	图像	abort error load
<AREA>	区域	mouseover mouseout
<BODY>…</BODY>	文档体	blur error focus load unload
<FRAMESET>…</FRAMESET>	帧集	blur error focus load unload
<FRAME>…</FRAME>	帧	blur focus
<FORM>…</FORM>	表单	submit reset
<INPUT TYPE="text">	文本框	blur focus change select
<TEXTAREA>…</TEXTAREA>	文本域	blur focus change select
<INPUT TYPE="submit">	提交	click
<INPUT TYPE="reset">	重置	click
<INPUT TYPE="radio">	单选	click
<INPUT TYPE="checkbox">	检查框	click
<SELECT>…</SELECT>	组合框	blur focus change

2.2.3　事件处理器

执行 JavaScript 代码的程序在事件发生时会对事件做出响应。为了响应一个特定事件而被执行的代码称为事件处理器。事件本身，例如 Click 事件，仅仅通知 JavaScript 该事件发生，以便执行事件处理器。事件处理器代码作为一个属性加在产生事件的 HTML 标签中。在 HTML 标签中使用事件处理器的语法是：

```
<HTML 标签  事件处理器="JavaScript 代码">
```

事件处理器的名称和事件同名，只是在前面加一个 on 前缀。例如，Click 事件的处理器是 onClick，Load 事件的处理器是 onLoad。回想一下 HTML 标签是大小写无关的，但 JavaScript 是大小写敏感的。因为事件处理器是 HTML 标签的一部分，所以它也是大小写无关的。因此，可以把 onClickEvent 事件处理器的名称写为 ONCLICK、onclick 或 Onclick。但是，通常习惯只把事件名称的第一个字母大写。

一个事件处理器的 JavaScript 代码包含在 JavaScript 事件处理器名称后的引号中。下面的代码使用<INPUT>标签创建一个普通的按钮，这个按钮和 OK 或 Cancel 按钮类似。这个标签同时也包含一个 onClick 事件处理器，它执行 JavaScript 内置的 alert()方法响应 Click 事件（事件在按钮被按下时触发）。需要注意的是 onClick 事件处理器执行的代码（alert()方法）包含在双引号中。

```
<INPUT TYPE="button" onClick="alert('You clicked a button!')">
```

JavaScript 内置的 alert()方法显示一个带有 OK 按钮的对话框。alert()方法需要有一个字符串或变量作为参数。注意因为上例程序的 alert()方法本身包含在双引号中，所以字符串应该括在单引号中。

alert()方法是上例程序中惟一的一条可执行语句。也可以同时执行几条语句，但是必须用分号隔开它们。下面这个例子中，事件处理程序包括两条语句，一条语句创建一个变量，另外一条语句使用 alert()方法显示这个变量。代码如下：

```
<INPUT TYPE="button" onClick="var message='You clicked a button'; alert(message)">
```

另外一个 JavaScript 内置的事件响应函数是 prompt()，它和 alert()有些类似。prompt()方法显示的对话框中包含一条消息、一个 OK 按钮和一个 Cancle 按钮。用户输入到 prompt()方法的文本框中的所有文本都赋给一个变量。prompt()方法的语法是 variable_name = prompt(message,default_text);。

下面的代码是 prompt()函数的一个例子，prompt()函数显示文本“ How old are you? ”。prompt()方法的第二个参数是默认文本“ Your Age ”，该文本显示在提示对话框的文本框中。用户按下 OK 按钮后，输入到文本框中的文本就赋给变量 yourAge，然后显示在一个警告

对话框中。如果用户没有修改默认文本就按下“ OK ”按钮，那么默认文本“ Your Age ”就赋给 yourAge 变量。如果用户按下“ Cancel ”按钮，就不会为 yourAge 变量赋值。

```
yourAge = prompt("How old are you?", "Your Age");
alert("Your age is " + yourAge);
```

下面要创建一个演示JavaScript的onLoad、onUnload、onClick和onChange事件的HTML文档，并且在文档中使用 alert()和 prompt()函数。

创建演示 JavaScript 事件的 HTML 文档：

1. 打开文本编辑器或 HTML 编辑器，创建一个新文档。

2. 输入起始<HTML>标签、带标题的<HEAD>标签、起始<SCRIPT>标签以及在不兼容浏览器中隐藏代码的 HTML 注释。

```
<HTML>
<HEAD>
<TITLE>JavaScript Events</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

3. 接着输入 var visitor_name = " "创建一个变量，以保存 Web 页面访问者的姓名。

4. 接着输入 function greet_visitor(){创建 greet_visitor()函数，greet_visitor()函数会在<BODY>标签中调用。

5. 输入以下代码，提示用户输入自己的姓名并把用户姓名赋给 visitor_name 变量。

```
visitor_name = prompt("Please enter your name",
      "Enter your name here");
```

6. 使用 alert()方法向来访者显示个性化的问候对话框，然后输入结束花括号。

```
alert("Welcome " + visitor_name + "!");
}
```

7. 接下来创建 farewell_visitor()函数，该函数被<BODY>标签中的 onLoad 事件调用。farewell_visitor()也使用了 visitor_name 变量。

```
function farewell_visitor() {
      alert("Thanks " + visitor_name +
            " for visiting this Web page!");
}
```

8. 输入如下代码结束<SCRIPT>和<HEAD>段。

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
```

9. 输入如下的<BODY>标签，它使用 onLoad 和 onUnload 事件调用 greet_visitor()函数和 farewell_visitor()函数。

```
<BODY onLoad="greet_visitor();"
onUnload="farewell_visitor();">
```

10. 输入<FORM>开始一个表单。

11. 创建如下两个<INPUT>标签，第一个<INPUT>标签创建一个包含 onChange 事件处理器的文本框，onChange 事件处理器在用户修改了文本框内容并把焦点移开文本框时显示一个 alert()对话框。第二个<INPUT>标签创建一个按钮，它使用文本框的 NAME 属性显示文本框的内容。

```
<INPUT TYPE="text" NAME="text_field" SIZE="25"
        onChange="alert(
                'The value of the text_field has changed.');"><BR>
<INPUT TYPE="button" VALUE="Display Text Field Contents"
        onClick="alert(text_field.value);">
```

12. 输入</FORM>结束表单。

13. 输入如下标签关闭<BODY>和<HTML>标签。

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

14. 把文件存为盘上 Tutorial.02 目录下的 GreetVisitor.html 文件，然后在 Web 浏览器中打开它。首先 onLoad 事件处理器被调用，执行 greet_visitor()函数显示一个提示对话框。图 2-11 显示的是提示对话框。

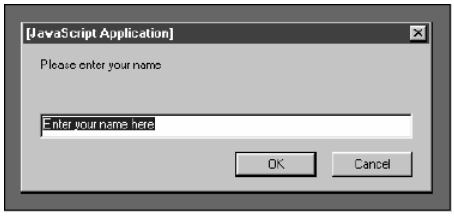


图 2-11　prompt()方法

15. 在对话框中输入您的姓名并按下“ OK ”按钮，警告对话框就会显示一条个性化的问候信息。单击警告对话框中的“ OK ”按钮，在文本框中输入“ Sample Text ”，然后按下“ Tab ”键把焦点移开文本域。因为修改了文本框中的文本，所以 onChange 事件处理器被调用，显示另外一个警告对话框。单击警告对话框中的“ OK ”按钮，然后单击“ Display Text Field Contents ”按钮调用 onClick 事件处理器，使用警告对话框显示文本框中的内容“ Sample Text ”。最后按下警告对话框的“ OK ”按钮。

16. 为了演示 onUnload 事件处理器，可以打开本章第一节中创建的 CompanyObjects.html 文件。在 Web 浏览器显示 CompanyObjects.html 文件之前，会调用 onUnload 事件处理器执行 farewell_visitor()函数，使用警告对话框向用户显示一条感谢信息。最后按下“ OK ”按钮关闭警告对话框。

17. 关闭 Web 浏览器窗口。

2.2.4 链接

回想一下 HTML 文档中的超文本链接，它可以用来打开文件或在 Web 上导航到其他文档。可以使用鼠标单击超文本链接来激活它。HTML 文档中的超文本链接通常有下划线，而且具有鲜明的颜色。未访问链接的默认颜色是蓝色，已访问链接的默认颜色是红色。超文本链接能够显示文件或 HTML 文档的实际名称和位置或者某些描述性的文本。其他类型的元素，例如图像，也能够链接到其他 HTML 文档、图像或文件上。在 HTML 文档中代表一个链接的文本或图像称为一个锚点。图 2-12 显示了一个包含几个锚点的 HTML 文档。

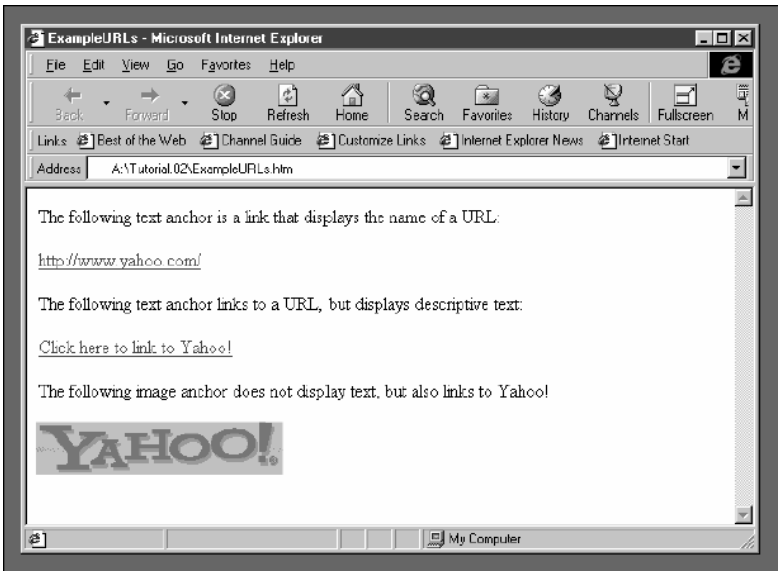


图 2-12 带锚点的 HTML 文档

HTML 文档使用<A>…</A>标签对（A 代表 anchor）创建一个链接。<A>标签的语法

是<A 属性>锚点文本或图像</A>。表 2-3 列出了<A>标签的一些属性。

表 2-3　常用<A>标签属性

属　　性	描　　述
NAME	锚点名称
HREF	被装载的文件、HTML 文档或图像的 URL
TITLE	被装载的文件、HTML 文档或图像的标题

锚点使用统一资源定位符（URL）指定一个 HTML 文档的名称和位置。HTML 文档中有两种类型的 URL：绝对的和相对的。绝对 URL 指的是一个特定的驱动器和目录或者 HTML 文档的完整 Web 地址。下面显示了一个“My Web Site”锚点，它包含一个 HTML 文档的绝对引用，该 HTML 文档是 www.MyWebSite.com 网站的 index.html。

```
<A HREF="http://www.MyWebSite.com/index.html">My Web Site</A>
```

一个绝对 URL 也可以指向本地计算机的一个文件，如下所示：

```
<A HREF="c:\MyWebPage\HomePage.html">My Web Site</A>
```

相对 URL 根据当前装载的 HTML 文档的位置指定文件的位置。相对 URL 用来装载与当前显示的 HTML 文档位于同一台计算机的 HTML 文档。如果当前显示的 HTML 文档位于 http://www.MyWebSite.com/WebPages 中，那么下面的相对 URL 会在 WebPages 目录中查找 AnotherWebPage.html 文件：

```
<A HREF="AnotherWebPage.html">Another Web Page</A>
```

也可以使用一个位于目前 Web 页所在目录的子目录的相对 URL，如下所示：

```
<A HREF="/MoreWebPages/YetAnotherWebPage.html">Yet Another Web Page</A>
```

如果所有的 HTML 文档都位于同一个目录，使用相对 URL 就非常方便，因为不需输入每个文件的完整路径。另外，如果重命名包含主 HTML 文档和链接文档的目录或者把它们移到其他计算机上，都不需更新相对 URL。例如，如果有一个包含 10 个链接到相同目录下 HTML 文档的主 HTML 文档，那么当把主 HTML 文档和 10 个链接文档移到一个新位置时不需更新相对 URL。但是，如果在创建这 10 个链接时使用绝对 URL，就必须更新每一个 URL 才能保证链接的正常使用。

提示：如果在 HTML 文档中使用<BASE>标签（为文档建立通用 URL）指定 URL，那么相对 URL 是相对于<BASE>标签中的 URL，而不是相对于文档自身的 URL 的。<BASE>标签通常和帧一起使用。

2.2.5 链接事件

链接使用的最主要的事件是 Click 事件。单击一个链接将自动执行 Click 事件，并打开链接相关的 URL。当用户单击一个链接时，Click 事件的执行由 Web 浏览器自动处理——不需要在<A>标签中添加 onClick 事件处理器。

但是，可能在某些情况下想使用自己的代码覆盖自动 Click 事件。例如，您可能希望提示用户一个特定链接将要打开的 HTML 文档的内容。当使用自己的代码覆盖自动 Click 事件时，需要在<A>标签中添加 onClick 事件处理器以执行自定义的代码。当使用自己的代码覆盖一个内部事件处理器时，必须使用 return 语句返回一个 true 或 false 值。值 true 表示希望 Web 浏览器继续并打开链接所指向的 URL。值 false 表示不希望 Web 浏览器打开这个链接。例如，图 2-13 中的<A>标签包括一个自定义的 onClick 事件处理器，onClick 事件处理器调用的 warnUser()函数返回一个由 confirm()方法产生的值。confirm()方法显示一个包含 Cancel 按钮和 OK 按钮的确认对话框。当用户单击确认对话框上的“ OK ”按钮时，返回值 true。当用户单击“ Cancle ”按钮时，返回 false。

```
<HTML>
<HEAD>
<TITLE>Custom OnClick event example</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--HIDE FROM INCOMPATIBLE BROWSERS
function warnUser()
return confirm(
"This link is only for people who love golden retrievers!");
}
//STOP HIDING FROM INCOMPATIBLE BROWSERS-->
</SCRIPT>
</HEAD>
<BODY>
<A HREF="GoldenRetrievers.html" onClick="return
warnUser();">Golden Retriever Club Home Page</A>
</BODY>
```

图 2-13 与定制 onClick 事件处理器的链接

注意图 2-13 中有两条 return 语句。warnUser()函数中的 return 语句把值返回给 onClick 事件处理器，onClick 事件处理器中的 return 语句把相同的值返回给 Web 浏览器。

链接使用的其他两个事件是 MouseOver 和 MouseOut 事件。鼠标移动经过一个链接时发生 MouseOver 事件，鼠标移开链接时发生 MouseOut 事件。MouseOver 和 MouseOut 事件最常见的用法之一是改变 Web 浏览器状态条中显示的文本。默认情况下，当鼠标经过一个

链接时，状态条中显示链接的 URL。但是，可以使用 onMouseOver 事件处理器在状态条中显示您自己为链接定制的消息。为了在状态条中显示自定义的消息，需要使用 JavaScript 的 status 属性。

onMouseOut 事件处理器通常用来在鼠标移开一个链接之后重置状态条中显示的文本。通常情况下，状态条中显示的文本使用语句 onMouseOut="status=' ';"清除，该语句赋给 status 属性一个空字符串。语句中的两个单引号指定一个空字符串。在语句中使用单引号而不是双引号是因为语句已经包含在一对双引号中了。（回忆一下，在双引号中不能再使用另外一对双引号）分号则作为 JavaScript 语句的结束标志。如果不想显示一个空字符串，也可以在状态条中显示另外一条自定义的消息。

下面的代码使用 onMouseOver 事件处理器在状态条中显示文本“ Golden Retriever Club Home Page ”来代替链接的 URL GoldenRetrievers.html。onMouseOut 事件处理器在鼠标移开链接后显示文本“ You almost visited the Golden Retriever Home Page! ”。

```
<A HREF="GoldenRetrievers.html" onMouseOver="status =
'Golden Retriever Club Home Page'; return true;"
onMouseOut="status = 'You almost visited the Golden
Retriever Home Page!';">Golden Retriever Club Home Page</A>
```

提示：在<SCRIPT>…</SCRIPT>标签对中，可以使用 defaultStatus 属性来指定鼠标不在一个链接之上时状态条中显示的默认文本。defaultStatus 属性的语法是 defaultStatus ="在这里输入默认文本。"；注意 defaultStatus 属性将覆盖 onMouseOut 事件处理器指定的任何文本。

注意前面例子中的 onMouseOver 事件处理器包含一个返回 true 值的 return 语句。和 onClick 事件处理器中的返回值不同，onMouseOver 事件处理器中的返回值 true 通知 Web 浏览器不要执行它自己在状态条中显示链接 URL 名称的事件处理程序。记住事件处理器的返回值并没有一致性，有些事件处理器需要一个返回值 true，而有些则需要 false。

下面创建两个演示链接的 Click、MouseOver 和 MouseOut 事件的 HTML 文档：

1. 启动文本编辑器或 HTML 编辑器并创建一个新文档。

2. 输入起始<HTML>标签、带标题的<HEAD>标签、起始<SCRIPT>标签以及在不兼容浏览器中隐藏代码的 HTML 注释。

```
<HTML>
<HEAD>
<TITLE>Red Page</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

3. 添加如下函数，该函数被链接的 onClick 事件处理器中调用。这个函数用于确认用

户是否希望打开链接指定的 URL。

```
function confirmPageChange() {
        return confirm(
"Are you sure you want to display the green page?");
}
```

4. 输入如下行结束<SCRIPT>和<HEAD>段。

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
```

5. 输入起始<BODY>标签，该标签包含 BGCOLOR 属性，此属性决定文档的背景色。

```
<BODY BGCOLOR="red">
```

6. 创建一个包含 onClick、onMouseOver 和 onMouseOut 事件处理器的链接。

```
<A HREF="GreenPage.html"
onClick="return confirmPageChange();"
onMouseOver=
"status = 'This link opens the green page'; return true;"
  onMouseOut=
"status ='You did not open the green page!!';">
Click here to open the green page</A>
```

7. 添加如下标签结束<BODY>和<HTML>标签。

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

8. 把文件存为盘上 Tutorial.02 目录下的 RedPage.html 文件，然后立刻把它另存为 GreenPage.html 文件。

9. 在新的 GreenPage.html 文件中，把<TITLE>标签中的文本改为 Green Page。

10. 在 confirmPageChange()函数中，该文本字符串中的单词 green 改为 red。

11. 把<BODY>标签中的 BGCOLOR 属性改为 green。

12. 在<A>标签中，把 HREF 由 GreenPage.html 改为 RedPage.html。同时把两个事件处理器中的单词 green 改为 red。注意因为 GreenPage.html 和 RedPage.html 位于相同的目录中，所以应使用相对 URL。

13. 保存文件，然后在 Web 浏览器中打开 RedPage.html 或 GreenPage.html。在每一个文件中单击链接会显示一个确认对话框，当单击“ OK ”按钮时会轮流显示另外一个文件，同时检查一下会发现每一个文件在发生 MouseOver 和 MouseOut 事件时会更新状态条文本。RedPage.html 和确认对话框的示例如图 2-14 所示。

图 2-14　RedPage.html 和确认对话框

14. 关闭 Web 浏览器窗口。

2.2.6　创建图像映射

图像映射由被分为区域的图像组成。每一个区域都使用一个<A>标签与一个 URL 相连，这些区域称为热点。使用鼠标单击热点可以打开与每个区域相连的 URL。可以使用<IMG>、<MAP>和<AREA>标签在 Web 页中创建图像映射。为了创建一个图像映射，必须在 Web 页中包含如下标签：

- ✧ 一个包含 SRC 和 NAME 属性的<IMG>标签，其中 SRC 属性指定图像文件的名称，NAME 属性则指定包含映射坐标的<MAP>…</MAP>标签对。
- ✧ 一个包含 NAME 属性的<MAP>…</MAP>标签对，该属性由<MAP>标签使用。
- ✧ 位于<MAP>…</MAP>标签对中的<AREA>标签，用于指定在图像内识别热点的坐标。

提示：图像映射有两种类型：服务器端图像映射和客户端图像映射。在服务器端图像映射中，映射图像每一个区域的代码位于服务器上，而客户端图像映射是 HTML 文档的一部分。本书涉及的是客户端图像映射。

当一个 HTML 文档中提交了用<IMG>标签指定的图像时，Web 浏览器会根据图像的高度和宽度在一个矩形区域中创建这个图像。图像所在矩形区域的尺寸用像素度量。像素（图形元素的简称，Picture Element）代表计算机屏幕上的一点。可以把像素想象为按照行列排

列在显示器中的上万或上百万的微小的点。像素的数目决定于计算机显示器的分辨率。例如，一个 VGA 显示器包含 480 行 640 列像素或大约 300000 个像素；一个 Super VGA 显示器包含 768 行 1024 列像素或大约 800000 个像素。

一个图像的像素使用 x 轴和 y 轴坐标引用，坐标开始于图像矩形区域的左上角，结束于右下角。图 2-15 显示了在一个 200 个像素高和 200 个像素宽的图像中像素是如何引用的。

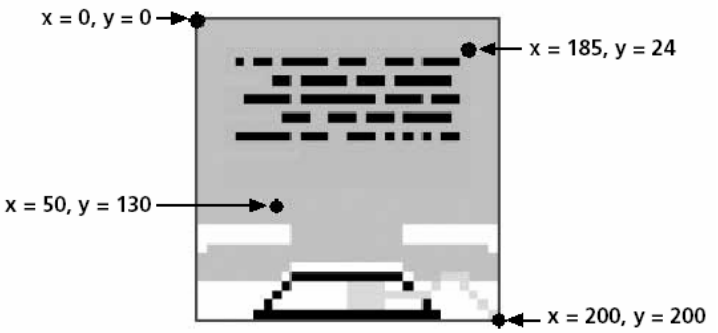


图 2-15 像素引用

<MAP>标签只有一个 NAME 属性，这个属性用来指定映射的名称。创建一个名为 imageMap 的<MAP>标签应该使用语句<MAP NAME="imageMap">。

<AREA>标签位于<MAP>…</MAP>标签对之间，它包含几个属性，如表 2-4 所示。

表 2-4 常用<AREA>标签属性

属 性	描 述
COORDS	以像素为单位的轮廓坐标，输入的坐标依赖于使用 SHAPE 属性指定的形状
HREF	与区域相关联的 URL
NAME	区域的名称
NOHREF	不与 URL 相关联的区域的占位符
SHAPE	定义区域的形状

当使用<AREA>标签在一个图像映射中定义一个热点区域时，使用 SHAPE 属性指定区域的形状并使用 COORDS 属性指定以像素为单位的轮廓坐标。SHAPE 属性可以设置为 CIRCLE、RECT（矩形）和 POLY（多边形）。每种形状的语法如下所示：

```
SHAPE=RECT COORDS="upper-left x, upper-left y,
      lower-right x, lower-right y"
SHAPE=CIRCLE COORDS="center-x, center-y, radius"
SHAPE=POLY COORDS="x1,y1, x2,y2, x3,y3,..."
```

在<IMG>标签提供的图像中使用一个图像映射时，需要使用 USEMAP 属性。图像映

射的名称前需要加一个#号，而且必须放在双引号中。下面的例子中使用<IMG>标签装载一个名为 sports.gif 的图像，同时使用一个名为 sports_map 的图像映射。

```
<IMG SRC="sports.gif " usemap="#sports_map">
```

和<A>标签相同，<AREA>标签也可以包括 onMouseOver 和 onMouseOut 事件处理器。图 2-16 显示了一个 HTML 文档示例，该文档创建一个有四个热点区域的图像映射，每个区域占矩形的四分之一。矩形的总尺寸是 250 个像素高，300 个像素宽。在热点区域上单击打开的 URL 由区域的<AREA>标签的 HREF 属性确定。每个区域的 onMouseOver 事件处理器在状态条中显示自定义的文本，onMouseOut 事件处理器则把状态条文本重置为空字符串。

图 2-17 显示了输出效果。

```
<HTML>
<HEAD>
<TITLE>Sports Map</TITLE>
</HEAD>
<BODY>
<IMG SRC="sports.gif" usemap="#sports_map">
<MAP NAME="sports_map">
<AREA HREF="baseball.html" SHAPE=rect coords="0,0,150,125"
        onMouseOver="status='Baseball Web Page.';
        return true;" onMouseOut="status=';return true;">
<AREA HREF="football.html" SHAPE=rect coords="150,0,300,125"
        onMouseOver="status='Football Web Page.';return true"
        onMouseOut="status=';return true;">
<AREA HREF="soccer.html" SHAPE=rect coords="0,125,150,250"
        onMouseOver="status='Soccer Web Page.';return true"
        onMouseOut="status=';return true;">
<AREA HREF="tennis.html" SHAPE=rect coords="150,125,300,250"
        onMouseOver="status='Tennis Web Page.';return true"
        onMouseOut="status=';return true;">
</MAP>
</BODY>
</HTML>
```

图 2-16　包含图像映射的 HTML 文档

接下来会创建本章开始时提到的北美的图像映射。Web 页第一次提交时装载的图像是一幅北美的地图，上面没有高亮度显示的国家。当移动鼠标经过图像时会高亮度显示每一个国家并显示国家名称。这些由一系列地图完成，每一个对应于一个北美国家。当鼠标经过一个国家时，一个 JavaScript 会临时改变所显示的图像。所需的图像位于盘上 Tutorial.02 目录中。

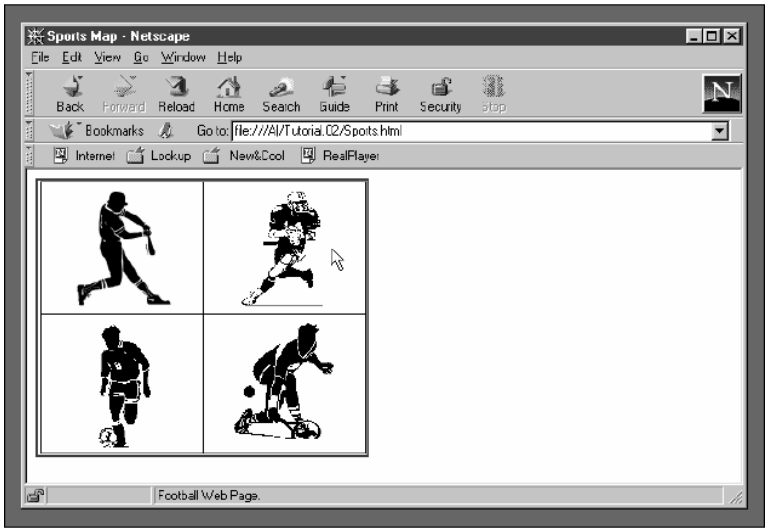


图 2-17　包含图像映射的 HTML 文档的输出

创建北美图像映射：

1. 首先，在盘上的 Tutorial.02 目录下创建一个名为 imageMap 的新目录。把 alaska.gif、canada.gif、grenland.gif、continential_us.gif、mexico.gif 和 north_america.gif 这六个图像从 Tutorial.02 目录拷贝到新的 imageMap 目录中。

2. 打开文本编辑器或 HTML 编辑器并创建一个新文档。

3. 输入起始<HTML>标签、带标题的<HEAD>标签、起始<SCRIPT>标签以及在不兼容浏览器中隐藏代码的 HTML 注释。

```
<HTML>
<HEAD>
<TITLE>North America</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

4. 输入如下的 change_image()函数。每一个 onMouseOver 事件处理器都调用这个函数来显示与每一个北美国家相关联的图像。每个 onMouseOver 事件处理器把所需的图像文件名传递给参数 image_name。语句 document.north_america.src = image_name;用来改变名为 north_ america 的<IMG>标签所显示的图像。将会在下面创建<IMG>标签 north_america。

```
function change_image(image_name) {
        document.north_america.src = image_name;
}
```

5. 创建如下函数在鼠标移开一个指定国家时把图像重置为 north_america.gif，该函数

被 onMouseOut 事件处理器调用。

```
function reset_image() {
    document.north_america.src = "north_america.gif";
}
```

6. 输入如下代码结束<SCRIPT>和<HEAD>段：

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
```

7. 输入起始<BODY>标签。

8. 输入如下代码创建图像并建立图像映射。您应该知道 onMouseOver 和 onMouseOut 事件处理器如何把必需的图像文件名传递给函数。注意 onClick 事件处理器是如何返回一个 false 值的。这防止用户直接打开每个区域关联的图像文件。同时注意<AREA>标签使用的是圆和多边形而不是矩形。

```
<img src="north_America.gif"
usemap="#north_America_map"NAME="north_America">
<MAP NAME="north_America_map">
<area shape=circle coords="44,46,20" HREF="alaska.gif"
onMouseOver="change_image('alaska.gif')"
onMouseOut="reset_image()" onClick="return false">
<area shape=poly coords="110,10,144,22,152,60,107,31"
HREF="greenland.gif"
onMouseOver="change_image('greenland.gif')"
onMouseOut="reset_image()" onClick="return false">
<area shape=poly coords=
"62,45,107,23,162,86,125,111,106,100,55,96,49,64"
HREF="canada.gif"
onMouseOver="change_image('canada.gif')" onMouseOut =
"reset_image()"onClick="return false">
<area shape=poly
coords="60,96,125,105,142,98,134,153,97,155,50,123"
HREF="continental_us.gif"
onMouseOver="change_image('continental_us.gif')"
onMouseOut="reset_image()" onClick="return false">
<area shape=poly
coords="61,135,122,165,109,181,65,159,60,136"
HREF="mexico.gif"
```

```
onMouseOver="change_image('mexico.gif')" onMouseOut =
"reset_image()" onClick="return false">
```

帮助：因为 onClick 事件使用返回的 false 值阻碍了超链接，所以<AREA>标签中的 HREF 属性没有做任何实质上的事情。但是如果没有包含 HREF 属性的话，onMouseOver 和 onMouseOut 事件在 Navigator 中都不起作用。

9. 输入如下标签结束<BODY>和<HTML>标签。

```
</MAP>
</BODY>
</HTML>
```

10. 把文件存为盘上 Tutorial.02 目录下的 ShowCountry.html 文件，然后在 Web 浏览器中打开它。测试程序并确保每个图像文件都被正确地装载。如果收到一个错误或者程序的功能不正确，检查一下是否正确输入了所有的起始和结束标签。同时确认 JavaScript 代码是否使用了正确的大小写。

11. 关闭浏览器窗口。

2.2.7 总结

- ✧ 事件是由 JavaScript 监视的特定情况。
- ✧ 用于创建与用户交互的输入域的<INPUT>标签可以产生事件。不同类型的<INPUT>标签元素产生不同的事件。
- ✧ 为了响应一个特定事件而被执行的代码称为事件处理器。
- ✧ 事件处理器代码作为一个属性加在产生事件的 HTML 标签中。
- ✧ 事件处理器的名称和事件同名，只是在前面加一个 on 前缀。例如，Click 事件的处理器是 onClick，Load 事件的处理器是 onLoad。
- ✧ JavaScript 内置的 alert()方法显示一个带有 OK 按钮的对话框。prompt()方法显示的对话框中包含一条消息、一个 OK 按钮和一个 Cancel 按钮。
- ✧ HTML 文档中有两种类型的 URL：绝对的和相对的。绝对 URL 指的是一个特定的驱动器和目录或者 HTML 文档的完整 Web 地址。相对 URL 根据当前装载的 HTML 文档的位置指定文件的位置。
- ✧ confirm()方法显示一个包含 Cancel 按钮和 OK 按钮的对话框。
- ✧ 鼠标移动经过一个链接时发生 MouseOver 事件，鼠标移开链接时发生 MouseOut 事件。
- ✧ 可以使用 JavaScript 的 status 属性在状态条中显示自定义的消息。
- ✧ 图像映射由被分为区域的图像组成。每一个区域都使用一个<A>标签与一个 URL 相连。

- ✧ 一个图像的像素使用 x 轴和 y 轴坐标引用，坐标开始于图像矩形区域的左上角，结束于右下角。
- ✧ 在<IMG>标签提供的图像中使用一个图像映射时，需要使用 USEMAP 属性。
- ✧ <AREA>标签可以包括 onMouseOver 和 onMouseOut 事件处理器。

2.2.8 问题

1. 一个_____ 或触发器，是由 JavaScript 监视的特定情况。

a. 通知
b. 事件
c. 警告
d. 提示

2. 当 Web 浏览器完成 HTML 文档装载时就会发生_____事件。

a. load
b. complete
c. display
d. click

3. <INPUT>标签_____用于创建与用户交互的输入域。

a. <INTERFACE>
b. <USERRESPONSE>
c. <BUTTON>
d. <INPUT>

4. 以下哪个选项是 Click 事件处理器正确的大小写？

a. OnClick
b. onCLICK
c. onClick
d. 事件处理器是大小写无关的

5. 下面哪个语句的语法是正确的？

a. onclick="alert('You Clicked a button!'); "
b. onclick="alert("You Clicked a button! ");"
c. onclick="alert(You Clicked a button!); "
d. onclick=alert('You Clicked a button!')。

6. 一个事件处理器中的多条 JavaScript 语句_____。

a. 包含在隔开的一系列圆括号中
b. 必须使用
标签隔开
c. 必须使用分号隔开
d. 不能在一个事件处理器中包含多条 JavaScript 语句

7. 以下哪个不是无效的 JavaScript 内置对话框函数？

a. confirm()

b. alert()

c. prompt()

d. message()

8. HTML 文档中用于代表一个链接的文本或图像被称为一个_____。

a. 占位符

b. 锚点

c. 挂钩

d. 链

9. 有两种 URL：绝对的和_____。

a. 静态的

b. 非绝对的

c. 永久的

d. 相对的

10. 有多个位于相同目录的 HTML 文档，它们彼此之间使用绝对 URL 链接。如果把这些文档移到其他 Web 站点，_____。

a. 必须手工更新每一个 URL

b. 不需更新 URL

c. 如果它们放在一个和原来相同的目录中，这些 URL 能够正常使用

d. 这些 URL 会自动更新自身以映射到新的 Web 站点

11. 下面哪个选项是取消链接单击事件的正确语法？

a. <A HREF="HomePage.html"onClick="return true">

b. <A HREF="HomePage.html"onClick="return false">

c. <A HREF="HomePage.html" onClick="return cancel">

d. <A HREF="HomePage.html" onClick="return override">

12. 使用 MouseOver 事件在状态条中显示“ Welcome to My Home Page ”，下面哪个选项使用了语法是正确的？

a. <A HREF="HomePage.html" onMouseOver="status= "Welcome to My Home Page; return true; ">Welcome to My Home Page</A>

b. <A HREF="HomePage.html" onMouseOver="status= true; ">Welcome to My Home Page</A>

c. <A HREF="HomePage.html"onMouseOver="status= 'Welcome to My Home Page'; return true; ">My Home Page</A>

d. <A HREF=”HomePage.html” onMouseOver=status= 'Welcome to My Home Page'; return true;>My Home Page</A>

13. 当鼠标没有位于一个链接之上时，显示在状态条中的默认文本由_____指定。

a. originalStatus

b. onMouseOutDefault

c. defaultText

d. defaultStatus

14. 下面哪个标签是图像映射不需要的？

a. <IMG>

b. <MAP>

c. <AREA>

d. <IMAGEMAP>

15. _____代表计算机屏幕上的一个单独的点。

a. 像素

b. 位

c. 单元

d. 元素

16. 下面哪种形状不是<AREA>标签的 SHAPE 属性的合法选项？

a. circle

b. rectangle

c. triangle

d. polygon

2.2.9　练习

1. 创建一个包含到您最喜欢的 Web 站点的链接列表的 HTML 文档。为每个链接创建一个函数，并使用链接的 onClick 事件处理器调用函数。每个函数都显示一个确认对话框询问用户是否真的想访问相关的 Web 页。把文件存为盘上 Tutorial.02 目录下的 ConfirmLinks.html 文件。

2. 创建一个政治调查 HTML 文档。在表单中使用文本域创建该调查，文本域包括用户政治面貌、所居住的州以及他们投票选举的总统、州长、参议员等政客的名字。使用 onLoad 事件处理器在状态条上显示字符串“这是一张在线政治调查表”。当用户输入每个域时，使用 onFocus 事件处理器在状态条上显示帮助信息。用户焦点离开文本域时，使用 onBlur 事件处理器显示一个包含他们在文本框中所输入信息的警告对话框。同时使用 onUnload 事件处理器显示一个包含“感谢您填写调查表”文本的警告对话框。把文件保存为盘上 Tutorial.02 目录下的 PoliticalSurvey.html 文件。

3. 使用文本“不要碰我”创建一个“发怒”链接。当移动鼠标经过链接时在状态条上显示文本“您做什么，不要碰我！”。使用 onMouseOut 事件处理器在状态条上显示文本

“谢谢您让我一个人呆着”。使用文档 ReallyMadLink.html 作为链接的 HREF 属性。把文件保存为盘上 Tutorial.02 目录下的 MadLink.html 文件。ReallyMadLink.html 文件使用 onLoad 事件处理器把文件背景色设为黑色。onLoad 事件处理器同时应该显示一个包含文本“现在您真的让我发怒了！”的警告对话框。ReallyMadLink.html 文件中也包含同样使用文本“不要碰我”的“发怒”链接。ReallyMadLink.html 文件的链接“不要碰我”的 onMouseOver 事件处理器在状态条上显示“除非您再次单击我，否则我不会平静的！”。单击这个链接将重新显示 MadLink.html 文件。

4. 使用图像处理程序，如 Paint，创建一个人形图像（如果您非常有艺术才能也可以创建更复杂的图形）。创建一个名为 BodyParts.html 的 HTML 文档，其中包含使用人形图像的图像映射。在人像身体的每一个部分创建热点。当移动鼠标经过每个身体部位时，在状态条中显示该部位的名称。同时在单击该链接时显示一个打印该部位名称的警告对话框。

5. 使用您家庭的一张扫描图片创建一个图像映射，并把它命名为 FamilyImageMap.html（如果没有扫描仪或数字相机，可以在 Internet 上搜索公众家庭照片以供练习使用）。文档中包含使用 defaultStatus 属性在状态条上显示您家庭名称的 onLoad 事件处理器。对照片中的每一个人创建一个热点，并在状态条中显示他或她的名字以及和您的关系。为每一个家庭成员创建个人 Web 页，当单击他或她的图像时打开该页面。在每个家庭成员的个人 Web 页中显示她或他的个人信息。最后在这些页面中包含一个返回 FamilyImageMap 页面的链接。

6. 使用 Paint 之类的图像处理程序创建一个包含动物名称的图像，然后在 Internet 公众剪辑图像库中搜索代表每个动物的图像。创建一个名为 ShowAnimal.html 的 HTML 文档。在 ShowAnimal.html 文件中，使用以前创建的包含动物名称的图像创建一个图像映射。使用动物名称作为热点，当移动鼠标经过热点时显示该动物的一幅图像。当鼠标没有位于动物名称之上时，显示一个包含文本“单击动物名称在这里显示其图像”的临时图片。

第 3 章　数据类型和运算符

案例

WebAdventure 的长期客户中有一家大银行 GlobalBank，它有一个复杂的 Web 站点。这家银行希望在 Web 站点的常规内容上增加交互功能以吸引用户使用他们的在线银行服务。到目前为止，WebAdventure 已经在 GlobalBank 的 Web 站点上增加了一个抵押计算器和一个汽车租赁计算器。最近，GlobalBank 的市场部门希望在 Web 站点上再增加一个简单的计算器，指导用户进行在线交易和账户结算。由于您快速掌握了 JavaScript，所以您的老板决定由您来负责开发这个计算器程序。

预览计算器程序

在本章中，将创建一个显示在线计算器的 JavaScript 计算器程序。为了创建这个计算器，必须学会如何使用变量、数据类型和运算符。

1. 在 Web 浏览器中打开盘上 Tutorial.03 目录中的 Tutorial3Calculator.html 文件，显示如图 3-1 所示的在线计算器。

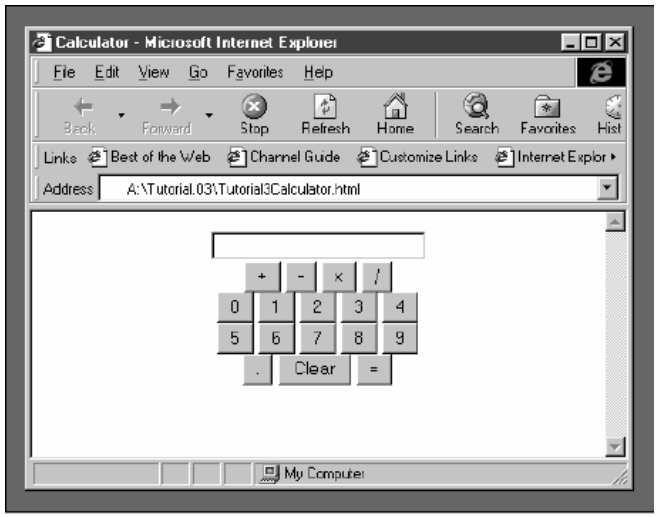


图 3-1　Tutorial3Calculator.html

2. 使用鼠标完成各种类型的计算，观察一下计算器是如何工作的。

3. 接下来，在文本编辑器或 HTML 编辑器中打开 Tutorial3Calculator.html 文件，分析一下代码。在该文档的主要部分，注意<INPUT>标签和 updateString()函数，它们结合在一起完成计算工作。

4. 分析完代码之后关闭文本编辑器或 HTML 编辑器。

3.1 使用数据类型和数组

本节目标

在本节您将学到：

- ✧ 如何使用数据类型
- ✧ 关于数字型数据类型
- ✧ 关于布尔值
- ✧ 如何使用字符串
- ✧ 如何使用数组

3.1.1 数据类型

变量中可以包含许多不同种类的值——例如事件、美元数目或一个人的名字。包含在 JavaScript 变量中的值或数据可以按照数据类型的类别进行分类。数据类型是一个变量所包含的信息的特定分类。只能赋给一个单独的值的数据类型称为原始数据类型。JavaScript 支持五种原始数据类型：整型数字、浮点型数字、布尔值、字符串和空值，如表 3-1 所示。

表 3-1 原始数据类型

数 据 类 型	描 述
整型数字	不包含小数的正数或负数
浮点型数字	包含小数的正数或负数，或者使用指数计数法书写的数字
布尔值	逻辑值 true 或 false
字符串	文本，如“ Hello World ”
空值	一个空的值

空值是一种数据类型，它同样也是一个能赋给变量的值。把值 null 赋给一个变量表明这个变量中不包含一个值。一个值为 null 的变量也有一个分配给它的值—— null 实际上就是这个值，它表示“ 没有值 ”。相比之下，一个未定义变量（第 2 章中学到的）永远不会有一个分配给它的值，或者说它没有声明，或不存在。

许多程序语言要求声明变量所包含的数据的类型。要求声明变量数据类型的编程语言称为强类型程序语言。因为数据类型在声明之后就不会再修改，所以强类型也称为静态类型。不要求声明变量数据类型的编程语言称为弱类型程序语言。因为数据类型在声明之后还可以改变，所以弱类型也称为动态类型。JavaScript 是一种弱类型程序语言。在 JavaScript 中不仅不需要声明变量的数据类型，而且也不允许这样做。JavaScript 的解释器会自动地判断变量中所存储数据的类型并相应地分配变量的数据类型。下面的代码演示了当变量分配了一个新值时变量的数据类型是如何自动改变的。

```
changingVariable = "Hello World";      //字符串
changingVariable = 8;                  //整型数
changingVariable = 5.367;              //浮点数
changingVariable = true;               //布尔量
changingVariable = null;               //空值
```

由于变量的数据类型在程序的执行过程中可以改变，所以当试图对字符串或空值变量执行算术运算时会出现一些问题。因此 JavaScript 引入了 typeof()运算符，使用它可以判断变量的数据类型。运算符可以用来操作一条语句的不同部分（在第 2 节会学习有关运算符的知识）。typeof()运算符的语法是 typeof(variablename);，typeof()运算符可以返回的值如表 3-2 所示。

表 3-2　typeof()运算符的返回值

返　回　值	描　　述
Number	整型或浮点数
String	文本字符串
Boolean	true 或 false
Object	对象、数组和空变量
Function	函数
Undefined	未定义变量

接下来创建一个分配不同数据类型给一个变量并打印该变量数据类型的程序。在程序中需要使用 typeof()运算符判断每一个变量的数据类型。

创建分配不同数据类型给一个变量并打印该变量数据类型的程序：

1. 启动文本编辑器或 HTML 编辑器，并创建一个新文档。
2. 输入<HTML>和文档的<HEAD>部分。

```
<HTML>
<HEAD>
<TITLE>Print Data Types</TITLE>
</HEAD>
```

提示：回想一下第 1 章，JavaScript 代码可以被放在<HEAD>或<BODY>段。在哪里放

置 JavaScript 代码是变化的，这决定于所写的程序。

3. 输入如下代码开始 HTML 文档体并创建一个预格式化文本容器：

```
<BODY>
<PRE>
```

4. 输入 JavaScript 段的起始语句。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

5. 输入语句 var differentType;声明一个名为 differentType 的变量。

6. 输入下面这行代码打印 differentType 变量的数据类型。因为变量还没有赋值，所以现在数据类型是未定义的。

```
document.writeln("The differentType variable is " + typeof(differentType));
```

7. 输入如下两行代码，分配给 differentType 变量一个字符串并重复执行数据类型打印语句。

```
differentType = "This is a text string";
document.writeln("The differentType variable is " + typeof(differentType));
```

8. 然后输入如下代码，改变 differentType 变量为整型、浮点型、布尔型和空数据类型。每次变量数据类型改变时都重复执行数据类型打印语句。

```
differentType = 100;
document.writeln("The differentType variable is "+ typeof(differentType));
differentType =3.679;
document.writeln("The differentType variable is "+ typeof(differentType));
differentType = false;
document.writeln("The differentType variable is "+ typeof(differentType));
differentType = null;
document.writeln("The differentType variable is "+ typeof(differentType));
```

9. 输入如下代码结束<SCRIPT>、<PRE>、<BODY>和<HTML>标签。

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -- >
</SCRIPT>
</PRE>
```

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

10. 保存文件为盘上 Tutorial.03 目录下的 PrintDataTypes.html。在 Web 浏览器中打开 PrintDataTypes.html 文件。应该看到与图 3-2 相同的行。

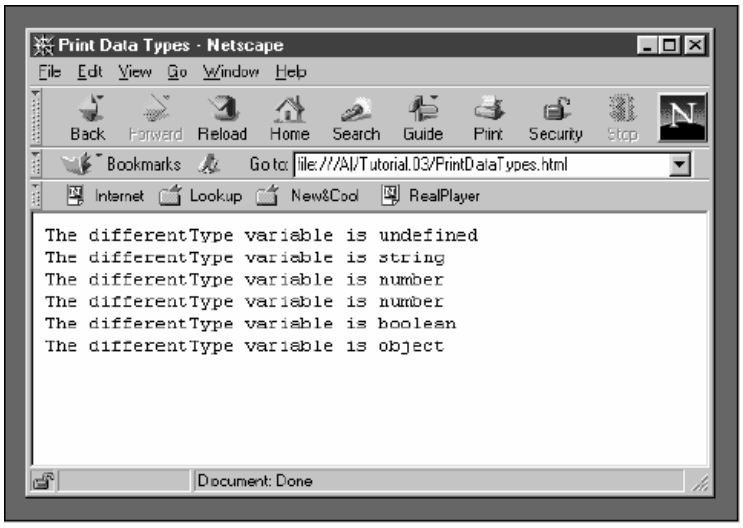


图 3-2　PrintDataTypes.html

11. 关闭 Web 浏览器窗口。

3.1.2　数字型数据类型

数字型数据类型在任何程序设计语言中都是一个重要部分，而且在进行算术运算时尤其有用。JavaScript 支持两种数字型数据类型：整型和浮点型数字。整型数字不包含小数位的正数或负数。JavaScript 中的整型数字的范围从 -9007199254740992（-2^{53}）到 9007199254740992（2^{53}）。数字-250、-13、0、2、6、10、100 和 10000 都是整型数字。数字-6.16、-4.4、3.17、0.52、10.5 和 2.7541 都不是整型数字；它们是浮点型数字，因为它们都包含小数位。浮点型数字包含小数位或者使用指数计数法书写。指数计数法，或科学计数法，是一种使用短格式书写大数或有许多小数位的数字的方法。使用指数计数法书写的数字由 1 到 10 之间的数值乘以 10 的次幂表示。数值 10 用大写或小写 E 代替。例如，200,000,000,000 使用指数计数法写作 2.0e11，表示“2 乘以 10 的 11 次方”。JavaScript 中的浮点型数字的范围大约从$\pm 1.7976931348623157\times 10^{308}$到$\pm 5\times 10^{-324}$。

提示：超过最大正值$\pm 1.7976931348623157\times 10^{308}$的浮点数产生一个特殊值 Infinity。超过最小负值$\pm 5\times 10^{-324}$的浮点数产生一个特殊值-Infinity。

下面创建一个为变量分配整型和指数型数字并打印变量值的程序：

1. 启动文本编辑器或 HTML 编辑器，并创建一个新文档。

2. 输入<HTML>和文档的<HEAD>部分。

```
<HTML>
<HEAD>
<TITLE>Print Numbers</TITLE>
</HEAD>
```

3. 输入如下代码开始 HTML 文档体并创建一个预格式化文本容器。

```
<BODY>
<PRE>
```

4. 输入 JavaScript 段的起始语句。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

5. 输入语句声明一个整型变量和一个浮点型变量。

```
var integerVar = 150;
var floatingPointVar = 3.0e7;
//浮点型数字 30000000
```

6. 输入下面这行代码打印变量。

```
document.writeln(integerVar);
document.writeln(floatingPointVar);
```

7. 输入如下代码结束<SCRIPT>、<PRE>、<BODY>和<HTML>标签。

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -- >
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

8. 保存文件为盘上 Tutorial.03 目录下的 PrintNumbers.html。在 Web 浏览器中打开 PrintNumbers.html 文件。整型数 150 和指数表达式 3.0e7 或数字 30,000,000 应该显示在浏览器窗口中。

9. 关闭 Web 浏览器窗口。

3.1.3　布尔值

布尔值是一个 true 或 false 的逻辑值。也可以把布尔值认为是 yes 或 no，或者 on 或 off。布尔值通常用来判定或者比较数据。在第 2 章中已经使用了布尔值来用自己的代码覆盖内部事件处理器。当使用自己的代码覆盖内部事件处理器时，需要使用 return 语句返回一个 true 或 false 值。可以同样使用确认对话框返回给事件处理器一个 true 或 false 值。用户单击确认对话框的 OK 按钮返回值 true，用户单击确认对话框的 Cancel 按钮返回值 false。图 3-3 显示了第 2 章看到的演示一个布尔值如何返回给事件处理器的程序。

```
<HTML>
<HEAD>
<TITLE>Custom onClick Event Example</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
function warnUser() {
        // The following line generates a Boolean value of true or
        // false and returns it to the event handler
        return confirm(
"This link is only for people who love golden retrievers!");
}
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
<BODY>
<A HREF = "GoldenRetrievers.html" onClick = "return
warnUser();">Golden Retriever Club Home Page</A>
</BODY>
</HTML>
```

图 3-3　返回一个布尔值给事件处理器的程序

在 JavaScript 程序设计中，只能使用单词 true 和 false 来表示布尔值。在其他程序设计语言中，可以使用整型值 1 和 0 来表示布尔值 true 和 false——1 表示 true，0 表示 false。JavaScript 在必要的时候可以把 true 和 false 转换为整数 1 和 0。例如，当试图在数学运算中使用布尔值 true 时，JavaScript 把变量转换为整数值 1。

提示：布尔值的名字来源于十九世纪的数学家 George Boole，他在发展数学逻辑方面做出了杰出贡献。

3.1.4 字符串

文本字符串包含用双引号或单引号括起来的零个或更多的字符。在一个程序中使用字符串的例子包括公司名称、用户名称、注释和其他类型的文本。文本字符串可以用作字面量或者把它们分配给一个变量。第一次使用字符串是在第 1 章的 document.write()和 document.writeln()函数中。变量可以分配一个称为空字符串的零长度字符串。例如，emptyVariable=" ";把空字符串分配给 emptyVariable 变量。空字符串是有效的字符串文字，它与空值和未定义值不同。

当在双引号括起来的字符串中包含一个引用字符串时，需要用单引号把引用字符串括起来。当在单引号括起来的字符串中包含一个引用字符串时，则需要用双引号把引用字符串括起来。不管使用哪种方法，字符串必须以相同类型的引号开始和结束。例如，document.write("This is a text string. ");是有效的，因为它是用双引号开始和结束的。语句 document.write("This is a text string. ');是无效的，因为它以双引号开始而以单引号结束。在这种情况下，会收到一条出错信息，因为 Web 浏览器不清楚字符串何时开始何时结束。图 3-4 显示的是一个打印文字字符串的程序的例子。图 3-5 显示了输出结果。

```
<HTML>
<HEAD>
<TITLE>Literal Strings</TITLE>
</HEAD>
<BODY>
<PRE>
<SCRIPT LANGUAGE="JavaScript1.2">
document.writeln("This is literal string");
document.writeln("This string contains 'quoted' string");
document.writeln('This is another example of "quoted" string');
var firstString = "This literal string has assigned to variable.";
var secondString = 'This literal string has also assigned to variable.';
document.writeln(firstString);
document.writeln(secondString);
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

图 3-4 LiteralStrings.html

提示：不像其他程序设计语言，JavaScript 中没有一个单独字符的专用数据类型，例如 C、C++和 Java 程序设计语言中的 char 数据类型。

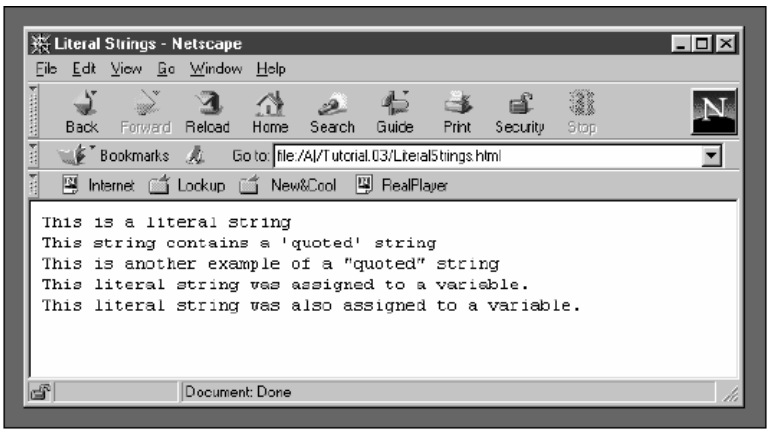


图 3-5　LiteralStrings.html 程序在 Web 浏览器中的输出

在程序中使用所有格和缩写时要特别注意单引号的使用，因为 JavaScript 解释器总是寻找第一个结束单引号或双引号以匹配一个起始单引号或双引号。例如，考虑下面的语句：

```
document.writeln('My City’s zip code is 01562.');
```

这条语句将引发一个错误。JavaScript 解释器假设文字字符串是以 City 后的单引号结束的，然后就会立刻在 City’s 后查找 document.writeln()函数的结束圆括号。为了解决这个问题，需要在 City’s 的单引号前添加一个转义字符。转义字符通知编译器或解释器跟随其后的字符有特殊的用途。在 JavaScript 中，转义字符是反斜杠（\）。在一个单引号前加上一个反斜杠通知 JavaScript 解释器按照普通键盘字符处理这个单引号，例如 a、b、1 或 2，而不是作为结束文本字符串的引号对的一部分。下面这条语句中的反斜杠通知 JavaScript 解释器把 City 后的单引号作为一个单引号打印。

```
document.writeln('My City\’s zip code is 01562.');
```

转义字符和其他字符一起使用可以在字符串中插入一个特殊字符。转义字符和其他字符一起使用的组合称为转义序列。反斜杠后跟单引号（\'）和反斜杠后跟双引号（\"）都是转义序列的例子。许多转义序列完成特殊的功能。例如转义序列\t 在字符串中插入一个制表符。表 3-3 中列出了 JavaScript 中一些可以加在字符串中的转义序列。

表 3-3　JavaScript 转义序列

转义序列	字　　符
\b	退格
\f	换页
\n	换行
\r	回车

续表

转义序列	字　　符
\t	水平制表符
\'	单引号
\"	双引号
\\	反斜杠

注意转义序列产生的字符中包括反斜杠。因为转义字符本身是一个反斜杠，所以必须使用转义序列“ \\ ”在字符串中包含一个反斜杠字符。例如，为了在一个字符串中包含路径“ C:\ WebPages\JavaScript_Files\ ”，必须在使用字符串中出现的每一个反斜杠处使用两个反斜杠字符，如下所示：

```
document.writeln("My JavaScript files are located in C:\\WebPages\\JavaScript_Files\\");
```

图 3-6 显示了一个使用了包含几个转义序列的字符串的程序示例。图 3-7 显示了输出结果。

```
<HTML>
<HEAD>
<TITLE>Escape Sequences</TITLE>
</HEAD>
<BODY>
<PRE>
<SCRIPT LANGUAGE="JavaScript1.2">
document.writeln("This line is printed \non two lines.");
        // New line
document.writeln("\tThis line includes a horizontal tab.");
        // Horizontal tab
document.writeln("My personal files are in c:\\personal.");
        // Backslash
document.writeln("My dog's name is \"Noah.\"");
        // Double quotation mark
document.writeln('Massachusetts\' capital is Boston.');
        // Single quotation mark
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

图 3-6　使用转义序列的字符串的程序

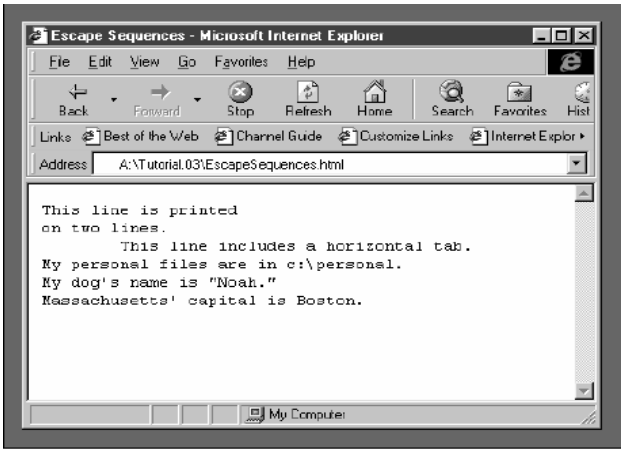


图 3-7　使用转义序列字符串程序的输出

```
<HTML>
<HEAD>
<TITLE>Strings with HTML tags</TITLE>
</HEAD>
<BODY>
<PRE>
<SCRIPT LANGUAGE="JavaScript1.2">
heading1_string = "<H1>Hello world (this is the H1 tag)</H1>";
document.writeln(heading1_string);
heading2_string = "<H2>This line is formatted with the H2 tag</H2>";
document.writeln(heading2_string);
italic_string = "<I>This line is italicized.</I>";
document.writeln(italic_string);
bold_string = "<B>This line is bolded.</B>”;
document.writeln(bold_string);
underlined_string = "<U>This line is underlined.</U>";
document.writeln(underlined_string);
formatted_string = "This line includes <I>italicized</I>,f
<B>bolded</B>, and <U>underlined</U> text.";
document.writeln(formatted_string);
hr_string = "Following this line is a horizontal rule.<HR>";
document.writeln(hr_string);
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

图 3-8　带 HTML 标签的字符串程序

字符串中不仅可以包括转义字符，还可以包括 HTML 标签。如果在 JavaScript 字符串

中包括 HTML 标签，它们必须位于字符串的起始和结束引号之间。例如，使用 document.write() 函数打印一个包括回车换行标签
的字符串，应该使用语句 document.write("There is a line break following this sentence.
");。HTML 标签不能在 JavaScript 代码中直接使用。因此，语句 document.write("There is a line break following this sentence. "
);会引发一个错误，因为
标签位于文字字符串之外。图 3-8 显示了一个带 HTML 标签的字符串程序示例。图 3-9 显示了输出结果。

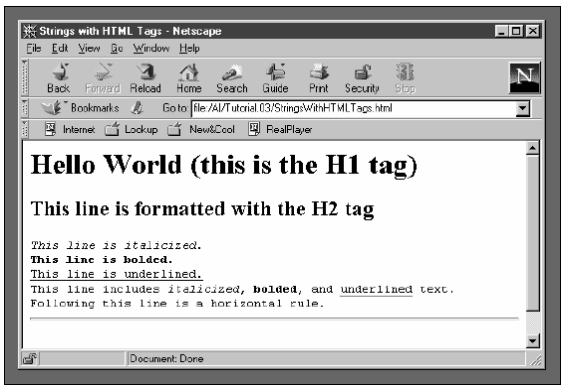


图 3-9　带 HTML 标签的字符串程序的输出

接下来将通过组合使用转义字符、HTML 标签和字符串，创建一个文件显示一个饭店每天的菜单。需要注意的是，可以只使用 HTML 标签轻松地创建相同的文档。这个练习的目的是演示文本字符串如何与 HTML 标签和转义字符组合使用。

创建组合使用转义字符、HTML 标签和字符串的文件：

1. 启动文本编辑器或 HTML 编辑器，并创建一个新文档。
2. 输入<HTML>和文档的<HEAD>部分。

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

3. 输入 JavaScript 段的起始语句。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

4. 声明如下变量并赋给文本、HTML 标签和转义字符组合起来的字符串。

```
var restaurant = "<H1>Small Town Restaurant</H1><BR>";
```

```
var specials = "<H2>Daily Specials for Wednesday</H2>";
var prixfixe = "<I>Prix fixe price:</I> <B>$9.95</B><HR>";
var appetizer = "<H3>Caesar Salad</H3>";
var entree = "<H3>Chef \'s    \"Surprise \"</H3>";
var dessert = "<H3>Chocolate Cheesecake</H3><HR>";
```

5. 接下来，输入如下语句打印这些变量：

```
document.write(restaurant);
document.write(specials);
document.write(prixfixe);
document.write(appetizer);
document.write(entree);
document.write(dessert);
```

提示：前面的语句使用的是 write()方法而不是 writeln()方法，这是因为字符串变量中所包含的标题层次风格自动强制换行。

6. 输入如下代码结束<SCRIPT>、<BODY>和<HTML>标签。

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -- >
</SCRIPT>
</BODY>
</HTML>
```

7. 把文件存为盘上 Tutorial.03 目录下的 DailySpecials.html 文件。在 Web 浏览器中打开 DailySpecials.html 文件。图 3-10 显示了输出结果。

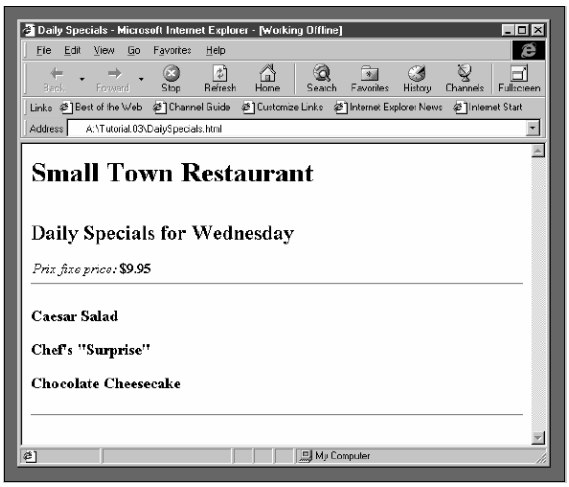


图 3-10　DailySpecials.html

8. 关闭 Web 浏览器窗口。

3.1.5 数组

一个数组包含由一个单独的变量名代表的一系列数据。可以把数组想象为包含在一个单独的变量中的变量的集合。数组是一个 JavaScript 对象。在第 2 章中，已经学过如何使用构造器函数和 new 关键字创建对象。因为数组是对象，所以可以使用 new 关键字和 JavaScript 的 Array()构造器对象创建数组。Array()对象和构造器函数是等价的，而且使用相同的语法创建，如下所示：

```
variable_name = new Array(元素个数);
```

注意 Array()构造器对象接收一个单独的参数表示包含在数组中的元素个数。数组中包含的每个数据称为一个元素。下面的代码创建一个有五个元素的数组 animals_list：

```
animals_list = new Array(5);
```

数组内元素的编号以索引号零（0）开始。引用数组中一个特定元素的方法是在数组名后跟用方括号包围的该元素的索引号。例如，animals_list 数组的第一个元素是 animals_list[0]，第二个元素是 animals_list[1]，第三个元素是 animals_list[2]，依此类推。除了必须包括数组元素的索引之外，为单独的数组元素赋值的方式与标准变量的赋值完全相同。下面的代码为 animals_list 数组的五个元素赋值：

```
animals_list[0] = "dog";          //第一个元素
animals_list[1] = "cat";          //第二个元素
animals_list[2] = "horse";        //第三个元素
animals_list[3] = "elephant";     //第四个元素
animals_list[4] = "llama";        //第五个元素
```

使用数组元素和使用其他类型变量的方法也是相同的。例如，下面的代码打印 animals_list 数组包含的五个元素的值：

```
document.writeln(animals_list[0]);      //打印 “ dog ”
document.writeln(animals_list[1]);      //打印 “ cat ”
document.writeln(animals_list[2]);      //打印 “ horse ”
document.writeln(animals_list[3]);      //打印 “ elephant ”
document.writeln(animals_list[4]);      //打印 “ llama ”
```

为一个数组元素赋值之后，可以在稍后改变它的值，就像修改程序中的其他变量一样。

为了把 animals_list 数组的第一个元素由 dog 改为 moose，在代码中需要包括语句“animals_list[0] = "moose";”，能够存储在数组元素中的变量数据类型并没有限制。例如，下面的代码创建一个数组并在这个数组的元素中存入多种数据类型：

```
multiple_types = new Array(5);
multiple_types[0] = "Hello World";      //字符串
multiple_types[1] = 10;                 //整型
multiple_types[2] = 3.156;              //浮点型
multiple_types[3] = true;               //布尔型
multiple_types[4] = null;               //空类型
```

使用 Array()构造器对象创建了一个新的数组对象时，声明数组元素的个数不是必须的，可以创建一个没有任何元素的数组，然后在需要时为数组添加新的元素。数组的大小可以动态改变。如果为数组中的一个未创建的元素赋值，这个元素会自动创建，同时创建在它之前的所有元素。例如，下面代码中的第一条语句创建一个没有任何元素的数组 animal_list，第二条语句把 aardvark 赋给第三个元素，执行这条语句时就会创建数组的前两个元素（animal_list[0]和 animal_list[1]）。

```
animal_list = new Array();
animal_list[2] = "aardvark";
```

提示：已经创建但未赋值的数组元素的初始值为空。

可以在创建数组时为数组元素赋值。下面的代码在创建 animal_list 数组时为它赋值，然后使用数组元素号打印每个值：

```
animals_list = new Array("dog", "cat", "horse");
document.writeln(animals_list [0 ]); // prints "dog"
document.writeln(animals_list [1 ]); // prints "cat"
document.writeln(animals_list [2 ]); // prints "horse"
```

接下来会创建一个数组，其中包含一年中的所有月份。

创建包含一年所有月份的数组：

1. 启动文本编辑器或 HTML 编辑器，并创建一个新文档。
2. 输入<HTML>和文档的<HEAD>部分。

```
<HTML>
<HEAD>
<TITLE>Months of the Year</TITLE>
</HEAD>
```

3. 输入如下代码开始 HTML 文档体并创建一个预格式化文本容器。

```
<BODY>
<PRE>
```

4. 输入 JavaScript 段的起始语句。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

5. 输入语句声明一个新的数组，其中包含 12 个元素。

```
var monthsOfYear = new Array(12);
```

6. 把一年中的 12 个月份赋给数组的 12 个元素。记住数组的第一个元素从 0 开始。因此，数组中包含一月的元素是 monthsOfYear[0]。

```
monthsOfYear [0 ] = "January";
monthsOfYear [1 ] = "February";
monthsOfYear [2 ] = "March";
monthsOfYear [3 ] = "April";
monthsOfYear [4 ] = "May";
monthsOfYear [5 ] = "June";
monthsOfYear [6 ] = "July";
monthsOfYear [7 ] = "August";
monthsOfYear [8 ] = "September";
monthsOfYear [9 ] = "October";
monthsOfYear [10 ] = "November";
monthsOfYear [11 ] = "December";
```

7. 接下来，输入如下语句打印数组的每一个元素。

```
document.writeln(monthsOfYear [0 ]);
document.writeln(monthsOfYear [1 ]);
document.writeln(monthsOfYear [2 ]);
document.writeln(monthsOfYear [3 ]);
document.writeln(monthsOfYear [4 ]);
document.writeln(monthsOfYear [5 ]);
document.writeln(monthsOfYear [6 ]);
document.writeln(monthsOfYear [7 ]);
document.writeln(monthsOfYear [8 ]);
```

```
document.writeln(monthsOfYear [9 ]);
document.writeln(monthsOfYear [10 ]);
document.writeln(monthsOfYear [11 ]);
```

提示：循环语句为打印数组所有元素提供了一种更为高效的方法。在第 4 章会学习有关循环语句的知识。

8. 输入如下代码结束<SCRIPT>、<PRE>、<BODY>和<HTML>标签。

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -- >
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

9. 保存文件为盘上 Tutorial.03 目录下的 MonthsOfYear.html。在 Web 浏览器中打开 MonthsOfYear.html 文件。图 3-11 显示了输出结果。

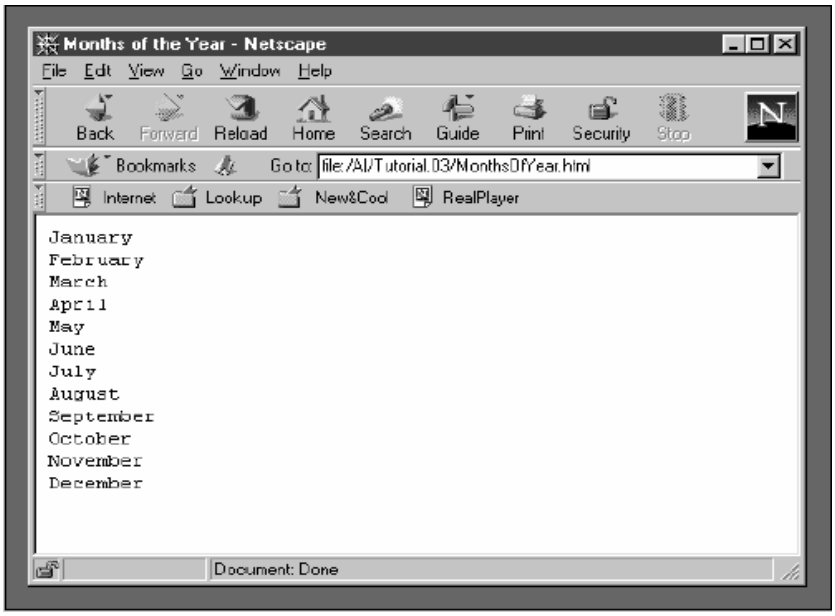


图 3-11　MonthsOfYear.html 的输出结果

10. 关闭 Web 浏览器窗口。

3.1.6　总结

✧　数据类型是一个变量所包含的信息的特定分类。

- ✧ 只能赋给一个单独的值的数据类型称为原始数据类型。
- ✧ 空值是一种数据类型，它同样也是一个能赋给变量的值。把值 null 赋给一个变量表明这个变量中不包含一个值。
- ✧ JavaScript 使用弱类型或动态类型而且声明变量时不需要数据类型。在 JavaScript 中，变量的数据类型在声明之后还可以改变。
- ✧ 整型数字是不包含小数位的正数或负数。
- ✧ 浮点型数字包含小数位或者使用指数计数法书写。指数计数法，或科学计数法，是一种使用短格式书写大数或有许多小数位的数字的方法。
- ✧ 布尔值是一个 true 或 false 的逻辑值。
- ✧ 文本字符串包含用双引号或单引号括起来的零个或更多的字符。由零个字符组成的字符串称为空字符串。
- ✧ 转义字符通知编译器或解释器跟随其后的字符有特殊的用途。转义字符和其他字符一起使用的组合称为转义序列。
- ✧ 一个数组包含由一个单独的变量名代表的一系列数据。数组中包含的每个数据称为一个元素。
- ✧ 可以使用 Array()构造器对象创建一个数组。Array()构造器对象接收一个单独的参数表示包含在数组中的元素个数。声明数组元素的个数不是必须的。
- ✧ 数组内元素的编号以索引号零（0）开始。
- ✧ 可以创建一个没有任何元素的数组，然后在需要时为数组添加新的元素。
- ✧ 数组的大小可以动态改变。
- ✧ 可以在创建数组时为数组元素赋值。

3.1.7 问题

1. 只能赋给一个单独的值的数据类型称为_____数据类型。

a. 简单
b. 基本
c. 原始
d. 单值

2. 以下哪个选项不是原始数据类型？

a. 字符串
b. 数字
c. 布尔值
d. 对象

3. 使用引号包围的文本是_____。

a. 文字字符串
b. 引用文本

c. 注释
d. 元素
4. 弱类型程序设计语言_____。
a. 不需要声明变量的数据类型
b. 需要声明变量的数据类型
c. 没有不同的数据类型
d. 没有变量
5. 可以使用_____判定变量的数据类型。
a. returnValue 函数
b. typeof()运算符
c. parseFloat()函数
d. toString 运算符
6. 以下哪个是整数？
a. 7.6
b. 12
c. 010
d. 0x1A
7. 以下哪个不是浮点数？
a. –439.35
b. 3.17
c. 10
d. –7e11
8. JavaScript 中的布尔值是逻辑值_____。
a. minimum 和 maximum
b. positive 和 negative
c. 1 和 0
d. true 和 false
9. 为了在包围在双引号中的字符串中包含双引号，以下哪条语句的语法是正确的？
a. "Some computers have \"artificial\"intelligence. "
b. "Some computers have " artificial" intelligence. "
c. "Some computers have /" artificial/" intelligence. "
d. "Some computers have ""artificial" " intelligence. "
10. 数组元素的编号从_____开始。
a. -1
b. 0
c. 1
d. 2

11. 以下哪个选项使用的是创建数组的正确语法？

a. variable_name = new Array;

b. variable_name = Array(元素个数);

c. variable_name = new Array(元素个数);

d. new Array(元素个数);

12. 以下哪个选项引用的是 employees[]数组的第一个元素？

a. employees[0]

b. employees[1]

c. employees[first]

d. employees[a]

3.1.8 练习

把以下练习创建的文件保存在盘上的 Tutorial.03 目录下。

1. 创建一个 HTML 文档显示您的工作简史，其中包括以前雇主的姓名、在每个公司的职位以及雇用的时间。在创建这个文档时，在字符串变量中包含所有 HTML 命令，同时使用 document.write()方法而不使用 document.writeln()方法。在文档中使用带转义字符或者 HTML 标签的字符串创建回车换行。把文档存为 WorkHistory.html。

2. 创建一个包含个人信息的 HTML 文档，其中包括您的姓名、年龄、您的汽车型号、房屋的抵押利率或汽车的贷款利率以及是否养狗。使用 write()或 writeln()方法打印每条信息。然后使用 typeof 运算符打印每项的数据类型信息，创建数据类型为字符串、数字和布尔型的项目。例如，姓名的数据类型是字符串，年龄是数字，而是否养狗的答案应该是布尔型，并把文档存为 PersonalInfo.html。

3. 创建一个包含您的简历的 HTML 文档。使用 write()方法建立标题部分，例如您的姓名、地址、以前雇主的姓名和雇用时间。在文本字符串中包含 HTML 标签和转义序列来格式化标题段。在文档体部分创建简历的主要段落。可能需要使用多个脚本段。把文档存为 Resume.html。

4. 创建一个 HTML 文档，在其中创建并打印一个包含所有您能想到的家庭成员的数组。数组应该包含在文档的<BODY>段的<SCRIPT>部分中。把文档存为 Family Array.html。

5. 创建一个 HTML 文档使用数组打印您最喜欢的 1990 年以来的歌曲，歌曲按照发表的时间排序。在文档中使用两个数组，并把它们放置在文档<BODY>段的<SCRIPT>部分中。第一个数组名称为 songs[]，其中包含 1990 年以来您所喜欢的歌曲列表。然后创建另外一个包含十个元素的数组，这十个元素中保存二十世纪 90 年代的每一个年份（1990、1991、1992 等）。第二个数组起名为 nineties[]。使用这两个数组打印每首歌曲，同时按照歌曲发表的时间排序。把文档存为 SongYears.html。

3.2　表达式和运算符

本节目标

在本节会学到：

- ✧ 如何使用表达式
- ✧ 如何使用算术、赋值、关系、逻辑和字符串运算符
- ✧ 如何创建计算器程序

3.2.1　表达式

变量和数据使用在表达式中会更为有用。表达式是字面量、变量、运算符和其他表达式的组合，JavaScript 解释器能够对它求值产生一个结果。JavaScript 解释器会把图 3-12 中的字面量和变量识别为表达式。

```
"this is a string variable"     // string literal expression
10                              // integer literal expression
3.156                           // floating-point literal expression
true                            // Boolean literal expression
null                            // null literal expression
employee_number                 // variable expression
```

图 3-12　文字串和变量表达式

可以使用操作数和运算符创建复杂的表达式。操作数是包含在表达式中的变量和文字串中，运算符就是使用在表达式中对操作数进行操作的符号。到目前为止已经使用了几个组合使用运算符和操作数的简单表达式。考虑下面的语句：

```
myNumber = 100;
```

这条语句就是一个表达式，它把数值 100 赋给 myNumber。这个表达式中的操作数是变量名 myNumber 和整型数值 100，等号（=）是赋值运算符。因为等号把表达式右边的值（100）赋给表达式左边的变量（myNumber），所以等号运算符是赋值运算符。表 3-4 列出了主要的 JavaScript 运算符类型。

提示：其他类型的 JavaScript 运算符还包括针对整数操作的位运算符，位运算符是一个复杂的话题。

表 3-4 JavaScript 运算符类型

运算符类型	描　　述
算术运算符	用于完成数学计算
赋值运算符	给变量赋值
关系运算符	比较操作数，返回一个布尔值
逻辑运算符	对布尔型操作数进行布尔操作
字符串运算符	操作字符串

JavaScript 运算符是二元或一元的。一个二元运算符要求在运算符前后各有一个操作数。语句 myNumber = 100;中的等号就是一个二元运算符。一元运算符只要求在运算符之前或之后有一个操作数，例如递增运算符（++），这是一个算术运算符，用来对操作数进行加 1 操作。语句 myNumber++改变 myNumber 的值为 101。

提示：运算符左边的操作数称为左操作数，而运算符右边的操作数称为右操作数。

接下来会学习不同类型的 JavaScript 运算符。

3.2.2 算术运算符

算术运算符用来在 JavaScript 中完成数学计算，比如加、减、乘、除。也可以返回计算结果的模数，即在使用一个数除另外一个数时剩下的余数。表 3-5 列出了 JavaScript 的二元算术运算符及其描述。图 3-13 是使用二元算术运算符的代码实例。

表 3-5 二元算术运算符

运　算　符	描　　述
+（加）	两个操作数相加
-（减）	两个操作数相减
*（乘）	两个操作数相乘
/（除）	两个操作数相除
%（求模）	两个操作数相除并返回余数

注意图 3-13，JavaScript 在赋值运算符的右边执行算术运算操作，然后把值赋给赋值运算符左边的变量。例如，在语句 returnValue = x + y;中，操作数 x 和 y 相加之后结果会赋给赋值运算符左边的变量 returnValue。

可以在赋值运算符的右边组合使用变量和字面量。例如，加法语句可以写为 returnValue = 100 + y;、returnValue = x + 200;或 returnValue = 100 + 200;。但是，左操作数中不能包含字面量，因为 JavaScript 解释器必须把返回值赋给一个变量。因此，语句 100 = x + y;将引

发一个错误。

```
var x,y,returnValue;
//ADDITION
x = 100;
y = 200;
returnValue = x + y;          //returnValue changes to 300
//SUBTRACTION
x = 10;
y = 7;
returnValue = x - y;          //returnValue changes to 3
//MULTIPLICATION
x = 2;
y = 6;
returnValue = x * y;          //returnValue changes 12
//DIVISION
x = 24;
y = 3;
returnValue = x / y;          //returnValue changes to 8
//NODULUS
x = 3;
y = 2;
returnValue = x % y;          //returnValue changes to 1
```

图 3-13　二元算术运算符示例

对字符串值进行算术运算时，JavaScript 解释器会试图把字符串值转换为数字。下面代码示例中的变量分配了字符串值而不是数字值，因为它们包含在双引号中。不过 JavaScript 解释器能够正确地执行乘法操作并返回一个值 6。

```
x = "2";
y = "3";
returnValue = x * y;              //返回值为 6
```

当使用加法运算符时，JavaScript 解释器不会把字符串转换为数字。当对字符串使用加法运算符时，字符串被组合在一起而不是相加。在下面的例子中，运算的返回结果是 23，这是因为变量 x 和 y 中包含的是字符串而不是数字：

```
x = "2";
```

```
y = "3";
returnValue = x + y;          //返回值为 23
```

使用一元运算符可以对一个单独的变量执行算术运算。表 3-6 列出了 JavaScript 中可用的一元算术运算符。

表 3-6 一元算术运算符

运 算 符	描 述
++（递增运算符）	操作数加 1
--（递减运算符）	操作数减 1
-（求反运算符）	返回操作数的相反数（负的或正的）

递增（++）或递减（--）运算符可以用作前缀或后缀运算符。前缀运算符放在变量之前，后缀运算符放在变量之后。++myVariable;和 myVariable++;语句都对 myVariable 加 1，但是这两条语句返回的值不同。递增运算法用作前缀运算符时，操作数的值在加 1 之后返回。递增运算符用作后缀运算符时，操作数的值在加 1 之前返回。类似地，递减运算法用作前缀运算符时，操作数的值在减 1 之后返回。递减运算符用作后缀运算符时，操作数的值在减 1 之前返回。如果想把递增或递减后的值赋给一个变量，那么使用前缀或后缀运算符就有一点区别。例如在下面的代码中，变量先使用前缀递增运算符加 1，然后把值赋给 newValue 变量：

```
var count = 10;
var newValue = ++count;     //newValue 所赋的值为 11
```

与递增运算符和递减运算符不同，求反运算符不能用作后缀运算符。求反运算符必须放在要改变为相反值的操作数之前。在下面的代码中，变量 x 开始被赋值为正数 10，然后使用求反运算符改变为-10。

```
var x = 10;
x = -x;              //x 改变为-10
```

下面会创建一个执行算术运算的程序。

创建执行算术运算的程序：

1. 启动文本编辑器或 HTML 编辑器，并创建一个新文档。
2. 输入<HTML>和文档的<HEAD>部分。

```
<HTML>
<HEAD>
<TITLE>Arithmetic Examples</TITLE>
```

```
</HEAD>
```

3. 输入如下代码开始 HTML 文档体并创建一个预格式化文本容器。

```
<BODY>
<PRE>
```

4. 输入 JavaScript 段的起始语句。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

5. 输入如下语句声明两个变量：一个数字型变量，将使用它来完成几个算术运算；一个结果变量，用来保存每次算术运算的结果。

```
var number = 100;
var result;
```

6. 然后输入如下语句对数字变量执行加、减、乘、除运算，并把结果赋给 result 变量。每次改变 result 变量时打印这个变量。

```
result = number + 50;
document.writeln("Result after addition = " + result);
result = number / 4;
document.writeln("Result after division = " + result);
result = number -25;
document.writeln("Result after subtraction = " + result);
result = number * 2;
document.writeln("Result after multiplication = " + result);
```

7. 接下来，输入如下两条语句。第一条语句使用递增运算符对 number 变量加 1 并把新值赋给 result 变量。第二条语句打印 result 变量。注意递增运算符用作前缀，所以赋给 result 变量的是新值。如果使用后缀递增运算符的话，赋给 result 变量的是旧值，即 number 变量加 1 前的值。

```
result = ++number;
document.writeln("Result after increment = " + result);
```

8. 输入如下代码结束<SCRIPT>、<PRE>、<BODY>和<HTML>标签。

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -- >
```

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

9. 保存文件为盘上 Tutorial.03 目录下的 ArithmeticExamples.html。在 Web 浏览器中打开 ArithmeticExamples.html 文件。图 3-14 显示了输出结果。

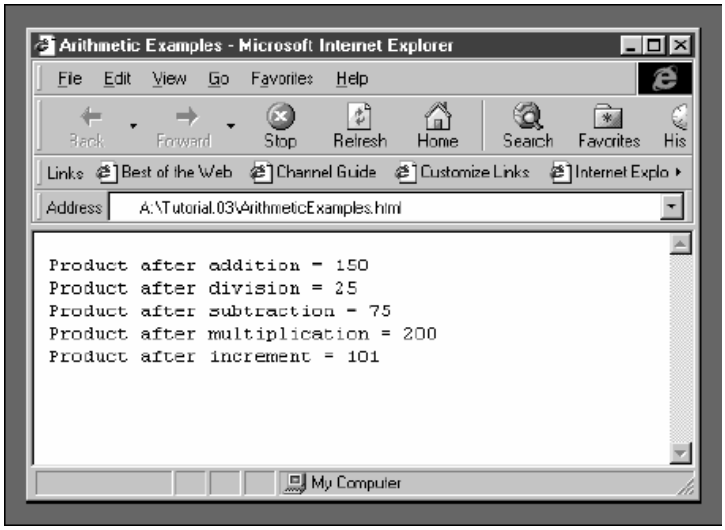


图 3-14 ArithmeticExamples.html 的输出结果

10. 关闭 Web 浏览器窗口。

3.2.3 赋值运算符

赋值运算符用来为变量赋值。已经使用过最常用的赋值运算符，即等号（=）为 var 语句中声明的变量赋值。等号为一个新变量赋一个初始值或者为已存在的变量赋一个新值。例如，下面的代码创建一个 myCar 变量，并使用等号为它分配一个初始值，然后使用等号再为它分配一个新值。

```
var myCar = "Ford";
myCar = "Corvette";
```

除等号外，JavaScript 还有其他一些赋值运算符。这些附加的赋值运算符对一个表达式中的变量和字面量执行数学计算，然后把新值赋给左操作数。表 3-7 中列出了最常用的 JavaScript 赋值运算符。

表 3-7　赋值运算符

运　算　符	描　　述
=	把右操作数赋给左操作数
+=	组合或相加右操作数和左操作数，并把结果赋给左操作数
-=	左操作数减去右操作数，并把结果赋给左操作数
*=	左操作数乘以右操作数，并把结果赋给左操作数
/=	左操作数除以右操作数，并把结果赋给左操作数
%=	左操作数除以右操作数，并把余数赋给左操作数

使用+=可以组合两个字符串或者累加两个数字。当对字符串进行操作时，运算符左边的字符串与运算符右边的字符串组合，并把新值赋给左操作数。在组合操作数之前，JavaScript 解释器会试图把非数字操作数（如字符串）转换为数字。如果一个非数字操作数不能被转换为数字，就会产生一个值 NaN。图 3-15 是不同赋值运算符的代码示例。

```
var x, y;

x = "Hello ";
x += "World";          // x changes to "Hello World"

x = 100;
y = 200;
x += y;                // x changes to 300

x = 10;
y = 7;
x  -  = y;             // x changes to 3

x = 2;
y = 6;
x *= y;                // x changes 12

x = 24;
y = 3;
x /= y;                // x changes to 8

x = 3;
y = 2;
x %= y;                // x changes to 1
```

图 3-15　赋值运算符示例

下面将创建一个使用赋值运算符的 HTML 文档：

1. 启动文本编辑器或 HTML 编辑器，并创建一个新文档。
2. 输入<HTML>和文档的<HEAD>部分。

```
<HTML>
<HEAD>
<TITLE>Assignment Examples</TITLE>
</HEAD>
```

3. 输入如下代码开始 HTML 文档体并创建一个预格式化文本容器。

```
<BODY>
<PRE>
```

4. 输入 JavaScript 段的起始语句。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

5. 输入如下语句对变量 changingVar 执行赋值运算，并在每次赋值运算之后打印结果。

```
var changingVar = "text string 1";
changingVar += " & text string 2";
document.writeln("Variable after addition assignment = "
        + changingVar);
changingVar = 100;
changingVar += 50;
document.writeln("Variable after addition assignment = "
        + changingVar);
changingVar -=-30;
document.writeln("Variable after subtraction assignment = "
        + changingVar);
changingVar /= 3;
document.writeln("Variable after division assignment = "
        + changingVar);
changingVar *= 8;
document.writeln("Variable after multiplication assignment = "
        + changingVar);
changingVar %= 300;
document.writeln("Variable after modulus assignment = "
```

```
+ changingVar);
```

6. 输入如下代码结束<SCRIPT>、<PRE>、<BODY>和<HTML>标签。

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -- >
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

7. 保存文件为盘上 Tutorial.03 目录下的 AssignmentExamples.html。在 Web 浏览器中打开 AssignmentExamples.html 文件。图 3-16 显示了输出结果。

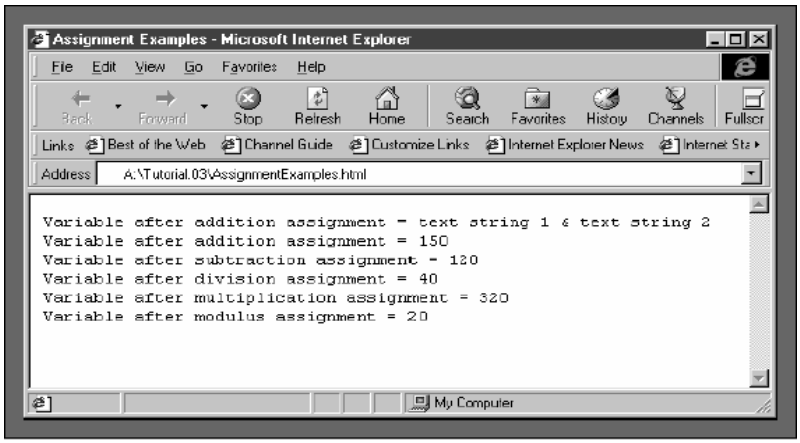


图 3-16　AssignmentExamples.html 的输出结果

8. 关闭 Web 浏览器窗口。

3.2.4　关系运算符

关系运算符用来比较两个变量是否相等，并判断一个数字值是否大于另一个。比较两个操作数后会返回一个布尔值 true 或 false。表 3-8 列出了 JavaScript 中的关系运算符。

表 3-8　关系运算符

运　算　符	描　　述
==（等于）	操作数相等返回 true
!=（不等于）	操作数不相等返回 true
>（大于）	左操作数大于右操作数返回 true
<（小于）	左操作数小于右操作数返回 true

续表

运 算 符	描 述
>=（大于等于）	左操作数大于或等于右操作数返回 true
<=（小于等于）	左操作数小于或等于右操作数返回 true

提示：关系运算符（==）由两个等号组成，它完成的功能和一个单独的等号（=）组成的赋值运算符不同。关系运算符比较数值，而赋值运算符分配数值。

数字或字符串值可以用作关系运算符的操作数。如果操作数是两个数字，JavaScript 解释器按照数值比较它们。例如，语句 returnValue = 5 > 4;返回 true，因为数字 5 在数值上大于数字 4。如果操作数使用的是非数字，JavaScript 解释器按照字母表顺序比较它们。语句 returnValue ="b">"a"；返回 true，因为按字母表顺序字母 b 比字母 a 大。如果一个操作数是数字另外一个是字符串，JavaScript 解释器会试图把字符串转换为数字。如果字符串不能转换为数字，就返回一个 false。例如，语句 returnValue = 10 =="ten";返回 false，因为 JavaScript 解释器不能把字符串“ ten ”转换为数字。图 3-17 是使用关系运算符的代码实例。

```
var x = 5, y = 6;
x == y;                                  // false
x != y;                                  // true
x > y;                                   // false
x < y;                                   // true
x >= y;                                  // false
x <= y;                                  // true
x = "text string";
y = "different string";
x != y;                                  // true
"abc" == "abc";                          // true
"abc" == "xyz";                          // false
```

图 3-17 关系运算符示例

下面将创建一个使用关系运算符的 HTML 文档：

1. 启动文本编辑器或 HTML 编辑器，并创建一个新文档。
2. 输入<HTML>和文档的<HEAD>部分。

```
<HTML>
<HEAD>
<TITLE>Comparison Examples</TITLE>
</HEAD>
```

3. 输入如下代码开始 HTML 文档体并创建一个预格式化文本容器。

```
<BODY>
<PRE>
```

4. 输入 JavaScript 段的起始语句。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

5. 输入如下语句对两个变量执行关系运算，把运算结果赋给变量 returnValue 并打印它。

```
var returnValue;
var Value1 = "first text string";
var Value2 = "second text string";
returnValue = Value1 == Value2;
document.writeln("Value1 equal to Value2: "
+ returnValue);
Value1 = 50;
Value2 = 75;
returnValue = Value1 == Value2;
document.writeln("Value1 equal to Value2: "
+ returnValue);
returnValue = Value1 != Value2;
document.writeln("Value1 not equal to Value2: "
+ returnValue);
returnValue = Value1 > Value2;
document.writeln("Value1 greater than Value2: "
+ returnValue);
returnValue = Value1 < Value2;
document.writeln("Value1 less than Value2: "
+ returnValue);
returnValue = Value1 >= Value2;
document.writeln(
"Value1 greater than or equal to Value2: "
+ returnValue);
returnValue = Value1 <= Value2;
```

```
document.writeln(
"Value1 less than or equal to Value2: "
        + returnValue);
```

6. 输入如下代码结束<SCRIPT>、<PRE>、<BODY>和<HTML>标签。

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -- >
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

7. 保存文件为盘上 Tutorial.03 目录下的 ComparisonExamples.html。在 Web 浏览器中打开 ComparisonExamples.html 文件。图 3-18 显示了输出结果。

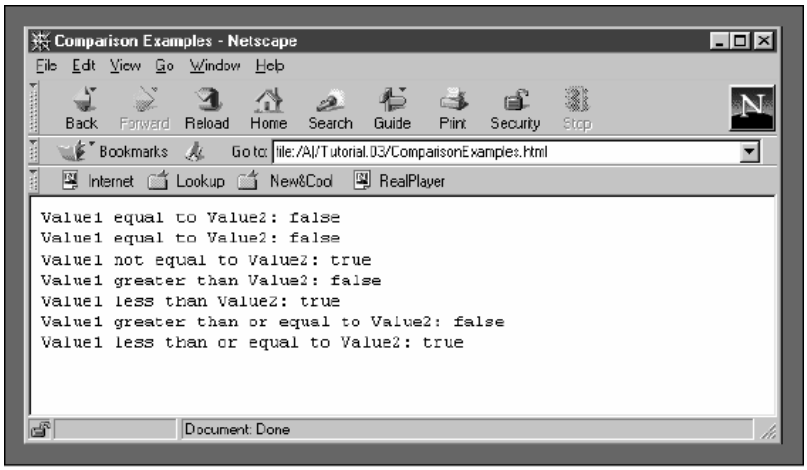


图 3-18 ComparisonExamples.html 的输出结果

8. 关闭 Web 浏览器窗口。

3.2.5 逻辑运算符

逻辑运算符用来比较两个布尔操作数是否相等。和关系运算符相同，比较两个操作数后会返回一个布尔值 true 或 false。表 3-9 列出了 JavaScript 中的逻辑运算符。

&&（与）和||（或）运算符都是二元运算符（需要两个操作数），！（非）运算符是一元运算符。逻辑运算符经常和关系运算符一起使用对表达式求值，这样可以把几个表达式的计算结果组合在一条语句中。例如，&&（与）运算符可以用来判断两个操作数是否返回相等的值，这些操作数本身通常就是表达式。

表 3-9　逻辑运算符

运　算　符	描　　述
&&（与）	左操作数和右操作数都是 true 就返回 true，否则返回 false
||（或）	左操作数或右操作数是 true 就返回 true，否则返回 false
!（非）	表达式为 false 返回 true，表达式为 true 返回 false

下面的代码使用&&运算符比较两个单独的表达式：

```
var a = 2; var b = 3;
var returnValue = a==2 && b==4;          //返回 false
```

逻辑或（||）语句用来检查两个表达式是否有一个为 true。例如，下面代码中的语句表示“ 如果 a 等于 2 或者 b 等于 3，returnValue 的值为 true，否则它的值为 false ”。

```
var a = 2; var b = 3;
var returnValue = a==2 || b==4;          //返回 true
```

尽管右操作数为 false，但是因为左操作数为 true，所以上面示例中的 returnValue 变量为 true。产生这种结果的原因是只要左操作数或右操作数中有一个为 true，||（或）运算符就返回 true。

下面的代码是!（非）运算符的示例，操作数为 false 返回 true，操作数为 true 返回 false。注意!（非）操作符是一元的，所以它只需一个操作数。

```
var x = true;
var returnValue = !x;          //返回 false
```

提示：逻辑运算符通常和条件或循环语句一同使用，例如 if else、for 和 while 语句。在第 4 章会学习关于条件和循环语句的知识。

下面将创建一个使用逻辑运算符的 HTML 文档：

1. 启动文本编辑器或 HTML 编辑器，并创建一个新文档。
2. 输入<HTML>和文档的<HEAD>部分。

```
<HTML>
<HEAD>
<TITLE>Logical Examples</TITLE>
</HEAD>
```

3. 输入如下代码开始 HTML 文档体并创建一个预格式化文本容器。

```
<BODY>
<PRE>
```

4. 输入 JavaScript 段的起始语句。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

5. 输入如下语句对两个变量使用逻辑运算符。

```
var trueValue = true;
var falseValue = false;
var returnValue;
document.writeln(!trueValue);
document.writeln(!falseValue);
document.writeln(trueValue || falseValue);
document.writeln(trueValue && falseValue);
```

6. 输入如下代码结束<SCRIPT>、<PRE>、<BODY>和<HTML>标签。

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -- >
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

7. 保存文件为盘上 Tutorial.03 目录下的 LogicalExamples.html。在 Web 浏览器中打开 LogicalExamples.html 文件。图 3-19 显示了输出结果。

8. 关闭 Web 浏览器窗口。

3.2.6 字符串运算符

JavaScript 中有两个运算符可以用于字符串：+和+=。加号和字符串一起使用时称为连接运算符。连接运算符（+）用来组合两个字符串。下面的代码组合一个字符串变量和一个文本字符串，并把结果赋给另外一个变量。

```
var firstString = "Ernest Hemingway wrote ";
var newString;
newString = firstString + "<I>For Whom the Bell Tolls</I>";
```

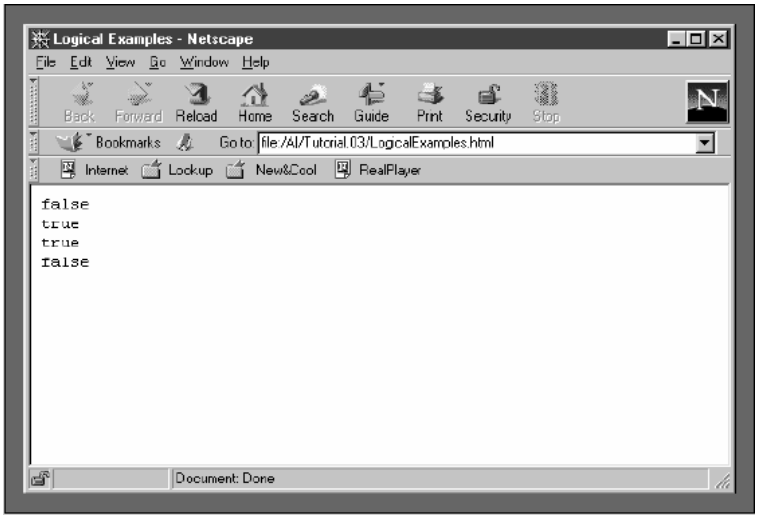


图 3-19　LogicalExamples.html 的输出结果

变量 firstString 和文字字符串组合结果赋给 newString 变量，它的值为“Ernest Hemingway wrote For Whom the Bell Tolls”。

也可以使用+=赋值运算符组合两个字符串。下面的代码同样组合两个文本字符串，但是没有使用 newString 变量。

```
var firstString = "Ernest Hemingway wrote ";
firstString += "<I>For Whom the Bell Tolls</I>";
```

注意加号可以用作连接运算符和加法运算符。与数字或包含数字的变量一起使用时，使用连接运算符的表达式返回两个数字的和。但是与字符串值和数字一起使用时，字符串值和数字将组合为一个新的字符串值，如下例：

```
var textString = "The legal voting age is ";
var votingAge = 18;
newString = textString + votingAge;
```

下面创建一个使用字符串运算符的 HTML 文档：

1. 启动文本编辑器或 HTML 编辑器，并创建一个新文档。
2. 输入<HTML>和文档的<HEAD>部分。

```
<HTML>
<HEAD>
<TITLE>String Examples</TITLE>
</HEAD>
```

3. 输入如下代码开始 HTML 文档体并创建一个预格式化文本容器。

```
<BODY>
<PRE>
```

4. 输入 JavaScript 段的起始语句。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

5. 输入如下使用字符串运算符的语句。在代码中使用您自己的姓名和出生地。

```
var name;
firstName = "your first name ";
lastName = "your last name ";
var placeOfBirth;
name = firstName + " ";
name += lastName;
placeOfBirth = "city where you were born ";
placeOfBirth += ",state where you were born ";
document.writeln("My name is " + name);
document.writeln("I was born in " + placeOfBirth);
```

6. 输入如下代码结束<SCRIPT>、<PRE>、<BODY>和<HTML>标签。

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -- >
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

7. 保存文件为盘上 Tutorial.03 目录下的 StringExamples.html。在 Web 浏览器中打开 StringExamples.html 文件。输出结果应该包括您的姓名和出生地。

8. 关闭 Web 浏览器窗口。

3.2.7 运算符优先级

在 JavaScript 中创建表达式时要注意运算符的优先级。运算符优先级是一个表达式中

运算符求值的优先顺序。表达式按照从左到右的原则求值，而且首先计算高优先级的运算符。JavaScript 中运算符的优先级顺序如下：

- ✧ 圆括号/方括号/点号（（）[] .）——优先级最高
- ✧ 求反/递增（!- ++ -- typeof void）
- ✧ 乘/除/求模（* / %）
- ✧ 加/减（+/-）
- ✧ 关系（< <= > >=）
- ✧ 相等（== !=）
- ✧ 逻辑与（&&）
- ✧ 逻辑或（||）
- ✧ 赋值运算符（= += -= *= /= %=）——优先级最低

提示：前面的优先级顺序列表中并没有包括所有的 JavaScript 运算符，只包括了本书涉及到的运算符。

语句 5 + 2 * 8 的值为 21，这是因为乘法运算符（*）的优先级高于加法运算符（+）。数字 2 和 8 首先相乘，结果为 16，然后再加上 5。如果加法运算符的优先级高于乘法运算符，那么这条语句的值应该是 56，因为 5 加 2 等于 7，然后再乘以 8。

可以看到，圆括号是具有最高优先级的运算符，所以圆括号可以用在表达式中改变表达式中单个运算符的求值顺序。例如，语句 5 + 2 * 8 的值为 21，但是写成（5 + 2）* 8 时它的值为 56。圆括号通知 JavaScript 解释器在乘以 8 之前把 5 和 2 相加。圆括号的使用使语句的值变为 56 而不是 21。

3.2.8　创建计算器程序

计算器程序使用一个特殊的转换函数 eval()函数完成计算。eval()函数对包含字符串的表达式进行求值。可以把字符串文字或字符串变量用作 eval()函数的参数。如果传递给 eval()函数的字符串文字或字符串变量不能求值，就会收到一个错误。语句 var returnValue = eval("5 + 3");返回数值 8 并把它赋给变量 returnValue。尽管包含在 eval()函数中的字符串并没有运算符，但是语句 var returnValue = eval("10");也可以正确地进行求值，并返回数值 10。eval()函数只有一个限制：不能传给它一个不包含运算符或数字的文本字符串。例如，语句 var returnValue = eval(" 'this is a text string' ");产生一个错误，因为它不包含数字或运算符。但是，语句 var returnValue = eval(" 'this is a text string' + 'this is another text string' ");能够正确地求值，因为传递给 eval()函数的字符串中包含连接运算符。

下面将创建 Calculator.html 程序。程序使用 inputString 变量包含算式中的操作数和运算符。一个算式赋给 inputString 后，使用 eval()函数执行运算。updateString()函数接受一个单独的代表一个数字或运算符的值。这个值使用赋值运算符+=加到 inputString 中。在 inputString 更新之后，它的值被赋给一个使用<INPUT>标签创建的名为 Input 的文本框。

Input 文本框是 Calculator 表单的一部分。Calculator 表单和 Input 文本框都作为 Document 对象的一部分调用。

提示：在第 6 章会学习有关表单的知识。

创建计算器程序：

1. 启动文本编辑器或 HTML 编辑器，并创建一个新文档。
2. 输入<HTML>和文档的<HEAD>部分。

```
<HTML>
<HEAD>
<TITLE>Calculator</TITLE>
```

3. 输入 JavaScript 段的起始语句。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

4. 输入语句 var inputString = " ";声明一个初始值为空字符串的 inputString 变量。
5. 接下来，输入下面的 updateString()函数，该函数用来更新 inputString 变量。

```
function updateString(value) {
        inputString += value;
        document.Calculator.Input.value = inputString;
}
```

6. 输入如下代码结束<SCRIPT>和<HEAD>标签。

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -- >
</SCRIPT>
</HEAD>
```

7. 添加<BODY>标签开始 HTML 文档的文档体部分。
8. 添加<CENTER>标签在页面里居中显示计算器。
9. 输入<FORM NAME="Calculato">开始 Calculator 表单。
10. 输入<INPUT TYPE="text" NAME="Input" SIZE="22">创建 Input 文本框。
11. 输入下面的 Input 标签创建代表计算器运算符的按钮。每一个标签都使用 onClick 方法传递给 updateString()函数一个值。

```
<BR>
<INPUT TYPE="button" NAME="plus"      VALUE=" + "
```

```
        onClick="updateString(' + ')">
<INPUT TYPE="button" NAME="minus"    VALUE=" - "
        onClick="updateString(' - ')">
<INPUT TYPE="button" NAME="times"    VALUE=" * "
        onClick="updateString(' * ')">
<INPUT TYPE="button" NAME="div"      VALUE=" / "
        onClick="updateString(' / ')"><BR>
```

帮助：使用“ button ”类型创建<INPUT>标签时，可以使用 VALUE 属性定义的标签中的空格和字符调整按钮宽度。为了调整每一个按钮的尺寸，计算器程序中创建的每一个按钮的标签的 VALUE 属性中都包含一些额外的空格。

12. 输入如下代表计算器数字的<INPUT>标签。

```
<BR>
<INPUT TYPE="button" NAME="zero"        VALUE=" 0 "
        onClick="updateString('0')">
<INPUT TYPE="button" NAME="one"         VALUE=" 1 "
        onClick="updateString('1')">
<INPUT TYPE="button" NAME="two"         VALUE=" 2 "
        onClick="updateString('2')">
<INPUT TYPE="button" NAME="three"       VALUE=" 3 "
        onClick="updateString('3')">
<INPUT TYPE="button" NAME="four"        VALUE=" 4 "
        onClick="updateString('4')">
<BR>
<INPUT TYPE="button" NAME="five"        VALUE=" 5 "
        onClick="updateString('5')">
<INPUT TYPE="button" NAME="six"         VALUE=" 6 "
        onClick="updateString('6')">
<INPUT TYPE="button" NAME="seven"       VALUE=" 7 "
        onClick="updateString('7')">
<INPUT TYPE="button" NAME="eight"       VALUE=" 8 "
        onClick="updateString('8')">
<INPUT TYPE="button" NAME="nine"        VALUE=" 9 "
        onClick="updateString('9')">
```

13. 输入代表小数点、清除按钮和 Calc 按钮的<INPUT>标签。注意 Calc 按钮的 onClick 事件使用 eval()函数对 inputString 变量执行计算。计算结果赋给 Input 文本框。

```
<BR>
<INPUT TYPE="button" NAME="point"    VALUE=" . "
```

```
        onClick="updateString('.')">
<INPUT TYPE="button" NAME="clear"    VALUE=" Clear "
        onClick="Input.value=''; inputString=''">
<INPUT TYPE="button" NAME="Calc"      VALUE=" = "
        onClick="Input.value=eval(inputString);">
```

14. 输入如下代码结束<FORM>、<CENTER>、<BODY>和<HTML>标签。

```
</FORM>
</CENTER>
</BODY>
</HTML>
```

15. 保存文件为盘上 Tutorial.03 目录下的 Calculator.html。在 Web 浏览器中打开 Calculator.html 文件。图 3-20 显示了输出结果。测试程序确保所有功能都能正确地执行。

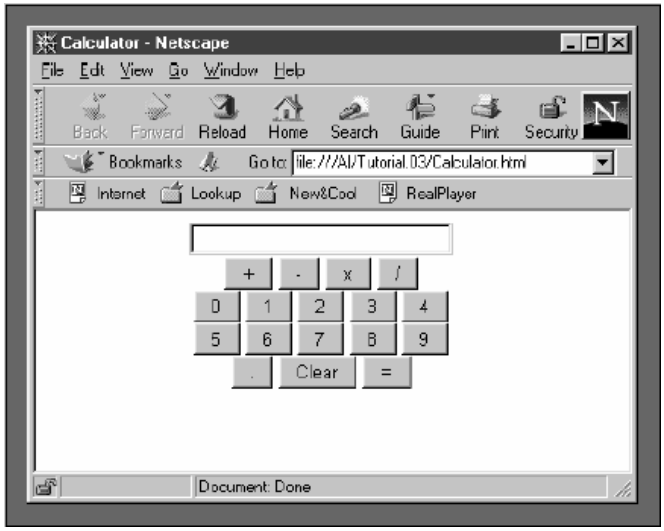


图 3-20　Calculator.html 的输出结果

16. 关闭 Web 浏览器窗口。

3.2.9　总结

- ✧ 表达式是字面量、变量、运算符和其他表达式的组合，JavaScript 解释器能够对它求值产生一个结果。
- ✧ 操作数是包含在表达式中的变量和文字串。
- ✧ 运算符就是使用在表达式中对操作数进行操作的符号。

- ✧ 一个二元运算符要求在运算符前后各有一个操作数。
- ✧ 一元运算符只要求在运算符之前或之后有一个操作数。
- ✧ 算术运算符用来在 JavaScript 中完成数学计算，如加、减、乘、除。
- ✧ 对字符串值进行算术运算时，JavaScript 解释器会试图把字符串值转换为数字。当使用加法运算符时，JavaScript 解释器不会把字符串转换为数字。
- ✧ 递增（++）或递减（--）运算符可以用作前缀或后缀运算符。前缀运算符放在变量之前，后缀运算符放在变量之后。
- ✧ 赋值运算符用来为变量赋值。
- ✧ 关系运算符用来比较两个变量是否相等，并判断一个数字值是否大于另一个。
- ✧ 逻辑运算符用来比较两个布尔操作数是否相等。
- ✧ 逻辑运算符经常和关系运算符一起使用对表达式求值，这样可以把几个表达式的计算结果组合在一条语句中。
- ✧ 加号，即加法运算符，与字符串一起使用时称为连接运算符。
- ✧ 运算符优先级是一个表达式中运算符求值的优先顺序。
- ✧ 圆括号可以用在表达式中改变表达式中单个运算符的求值顺序。
- ✧ eval()函数对包含字符串的表达式进行求值。

3.2.10　问题

1. 要求在运算符前后各有一个操作数的运算符称为_____运算符。

a. 一元

b. 二元

c. 双重

d. 复合

2. 求模运算符（%）_____。

a. 基于 16（十六进制）格式转换操作数

b. 返回操作数的绝对值

c. 计算一个操作数相比于另外一个操作数的百分比

d. 把两个数相除并返回余数

3. 假设变量 count 的值为 10，语句 returnValue = count++;中赋给 returnValue 变量的值是多少？

a. 10

b. 11

c. 12

d. 20

4. 假设变量 returnValue 包含字符串值“ 50 Main Street ”，语句 returnValue += "100"; 中赋给 returnValue 变量的值是多少？

a. 150

b. 50 Main Street 100

c. 100

d. 100 Main Street 50

5. 语句 returnValue = "First String" =="Second String";中赋给 returnValue 变量的值是多少？

a. First String

b. Second String

c. true

d. false

6. 语句 returnValue = 100 != 200;中赋给 returnValue 变量的值是多少？

a. 100

b. 200

c. true

d. false

7. 语句 returnValue = 50 != "fifty";中赋给 returnValue 变量的值是多少？

a. true

b. false

c. 50

d. “ fifty ”

8. 如果 _____ ，&&（与）运算符返回 true。

a. 左操作数返回 true

b. 右操作数返回 true

c. 左操作数和右操作数都返回 true

d. 左操作数和右操作数都返回 false

9. 左操作数或右操作数返回 true 时返回 true 的运算符是_____。

a. ||

b. ==

c. %%

d. &&

10. 假设 x 的值为 true，语句 returnValue += "100";中赋给 returnValue 变量的值是多少？

a. true

b. false

c. null

d. undefined

11. _____运算符用于组合两个字符串。

a. 联合

b. 结合
c. 组合
d. 连接

12. 表达式中运算符求值的优先顺序称为 _____ 。

a. 特权优先级
b. 运算符优先级
c. 表达式求值
d. 优先级求值

13. JavaScript 中具有最高优先级的运算符是 _____ 。

a. 赋值运算符
b. 加/减运算符
c. 关系运算符
d. 圆括号（）

14. 表达式 4*(2 + 3)的值是多少？

a. 11
b. -11
c. 20
d. 14

15. 假设 x 的值为 1，语句 returnValue = eval(x + "2 * 2");中赋给 returnValue 变量的值是多少？

a. "1 + 2 * 2"
b. "12 * 2"
c. 6
d. 24

3.2.11　练习

1. 创建一个计算房间所需地毯尺寸的 HTML 文档，其中包括三个文本框。以英寸为单位创建一个房间宽度文本框和一个房间长度文本框，同时创建一个每平方英寸的地毯费用的文本框。计算成本时，需要在为壁橱和房间内的其他东西在平方英寸总数上加 25%。使用警告对话框显示总成本。把文件存为盘上 Tutorial.03 目录下的 CarpetCost.html 文件。

2. 下面的表达式中赋给 returnValue 变量的值是多少？

a. returnValue = 2 == 3;
b. returnValue = "2" + "3";
c. returnValue = 2 >= 3;
d. returnValue = 2 <= 3;
e. returnValue = 2 + 3;

f. returnValue = (2 >= 3) && (2 > 3);

g. returnValue = (2 >=3) || && (2 > 3);

3. 创建一个 HTML 文档，在文档的<HEAD>段中包含的<SCRIPT>段声明五个全局变量：name、age、monthOfBirth、dataOfBirth 和 yearOfBirth。在文档的<BODY>段中包含的另一个<SCRIPT>段声明另外一个名为 birthInfo 的变量。使用+=赋值运算符把所有的五个全局变量组合到 brithInfo 变量中，然后使用 document.write()方法打印 brithInfo 变量。把文件存为盘上 Tutorial.03 目录下的 BirthInfo.html 文件。接下来，打开 HTML 文档察看结果。您对结果满意吗？如何进行格式和字符串连接的方式。

4. 创建一个温度转换程序计算器，把华氏度转换为摄氏度，摄氏度转换为华氏度。华氏度温度减去 32，然后乘以 0.55，就可以把华氏度转换为摄氏度。摄氏度温度乘以 1.8，然后加上 32，就可以把摄氏度转换为华氏度。把文件存为盘上 Tutorial.03 目录下的 ConvertTemperature.html 文件。

5. 使用圆括号改变下面代码的优先级顺序，使最后结果为 581.25（目前这种语法产生的结果值 x 是 637.5）。把文件存为盘上 Tutorial.03 目录下的 ImprovedProgram.html 文件。

```
var x = 75;
x = x + 30 * x /4;
```

6. 使用搜索条件“ JavaScript calculat ”在 Internet 搜索引擎如 Yahoo!、HotBot、Lycos 或 AltaVista 上搜索 JavaScript 计算的示例（搜索 calculat 会返回包括单词 calculator 或 calculation 和其他相似条件的实例）。访问几个包含 JavaScript 计算工具的 Web 站点，察看其中的 HTML 和 JavaScript 代码。写一篇一页的文章描述一下创建计算器的技术以及其他类型的完成数学计算功能的 JavaScript 工具。

第 4 章　使用控制结构和语句进行流程控制

案例

Cartoon and Animation Warehouse 通过他们的 Web 站点销售卡通和动画电影录像片。该公司希望 WebAdventure 为他们的 Web 站点寻找吸引新业务的方法。极富创造力的 WebAdventure 员工建议创办一个有奖竞赛。Web 站点的访问者可以做一个简单的测验，测试一下他们的卡通和动画电影知识。答对所有问题的人可以得到一件印有他们喜欢的卡通人物的免费 T 恤。于是您的老板要求您创建一个为客户演示的测验程序原型。

预览 CartoonQuiz.html 文件

在本章中，会创建一个测试用户的卡通和动画电影知识的 JavaScript 测验程序。将使用不同的控制结构和语句创建几个版本的 CartoonQuiz 程序。

1. 在 Web 浏览器中打开盘上 Tutorial.04 目录中的 CartoonQuiz.html 文件。图 4-1 在 Web 浏览器中显示了一个该程序的示例。

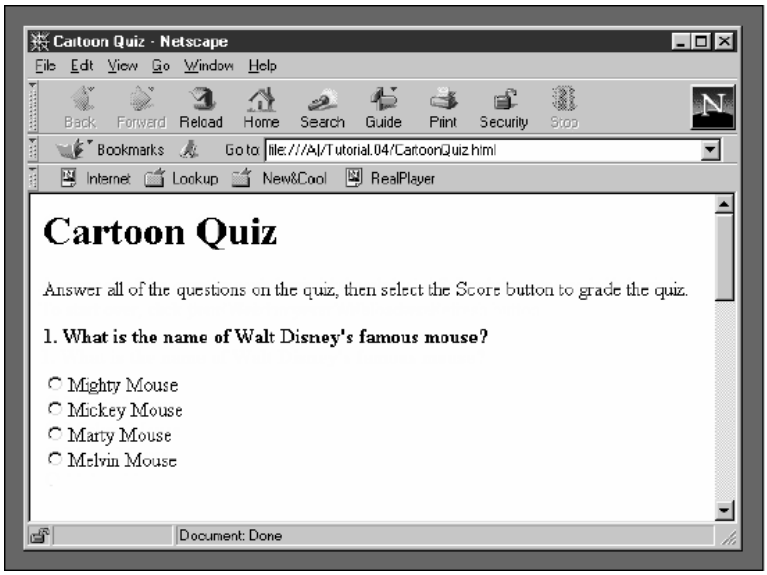


图 4-1　卡通测验

2. 回答测验中的问题，然后单击页面底部的“ Score ”按钮看看您做的怎么样。

3. 完成之后关闭浏览器窗口。

4. 接下来，在文本编辑器或 HTML 编辑器中打开 CratoonQuiz.html 文件，分析一下代码。注意以 for 和 if 开始的语句。这些语句是在本章将要创建的一些控制结构的实例。每一个问题中的多个选项使用<FORM>…</FORM>标签对中的<INPUT>标签创建。

5. 分析完代码之后关闭文本编辑器或 HTML 编辑器。

4.1 判　　断

本节目标

在本节将学会如何使用：

- ✧ if 语句
- ✧ if…else 语句
- ✧ 嵌套 if 语句
- ✧ switch 语句

4.1.1 if 语句

当编写一个计算机程序时，不考虑程序设计语言，经常需要根据某些预定条件执行不同的语句组。例如，可能需要根据一天中的时间或者运行程序的浏览器类型执行不同的语句组。另外，可能需要根据用户的输入执行不同的语句组。举例来说，有一个让用户发出在线订单的 Web 页面，如果一个用户单击“ Add to Shopping Cart ”按钮，就必须执行一组用于建立用户购买物品列表的语句。但是，如果用户单击“ Checkout ”按钮，就必须执行另外一组完全不同的完成交易的语句。决定一个程序中语句执行顺序的过程称为判断或流程控制。用于判断的特定类型的 JavaScript 语句称为判断结构。

控制程序流程最常用的方法之一是使用 if 语句。if 语句用来在一个条件表达式返回 true 值时执行特定的程序代码。if 语句的语法如下：

```
if (条件表达式){
        语句(s);
}
```

if 语句包括三个部分：关键字 if、包含在圆括号中的条件表达式和可执行语句。注意条件表达式必须包含在圆括号中。

如果 if 语句中的条件表达式的值为 true，那么紧随 if 关键字和条件之后的语句（语句

组）就被执行。if 语句执行之后，所有随后的代码都正常地执行。考虑一下图 4-2 所示的例子。if 语句使用等号（==）关系运算符判断 exampleVar 是不是等于 5。因为条件值为 true，所以会显示两个警告对话框。第一个警告对话框是在条件值为 true 时由 if 语句产生的，第二个警告对话框在 if 语句完成之后执行。

```
var exampleVar = 5;
if (exampleVar == 5)          // CONDITION EVALUATES TO 'TRUE'
        alert("The variable is equal to '5'.");
alert("This dialog box is generated after the if statement.");
```

图 4-2　一个条件值为 true 的 if 语句

提示：图 4-2 中紧随 if 语句的语句可以和 if 语句本身写在同一行。但是，使用换行和缩进可以使阅读更加容易。

```
var exampleVar = 4;
if (exampleVar == 5)          // CONDITION EVALUATES TO 'FALSE'
        alert("This dialog box will not be displayed.");
alert("This is the only dialog box displayed.");
```

图 4-3　一个条件值为 false 的 if 语句

相比之下，图 4-3 中的代码仅显示第二个警告对话框。因为赋给 exampleVar 的值是 4 而不是 5，所以条件值为 false。

可以使用由多条 if 语句组成的命令块创建一个判断结构。命令块指的是包含在一对花括号中的多条语句，它和包含在花括号中的函数语句类似。每个命令块有一个起始花括号（{）和一个结束花括号（}）。如果一个命令块中缺少了起始或结束花括号，就会发生错误。图 4-4 显示的程序在 if 语句的条件表达式求值为 true 时运行一个命令块。

```
var exampleVar = 5;
if (exampleVar == 5) {        // CONDITION EVALUATES TO 'TRUE'
        document.writeln("The condition evaluates to true.");
        document.writeln("exampleVar is equal to 5.");
        document.writeln("Each of these lines will be printed.");
}
document.writeln(
        "This statement always executes after the if statement.");
```

图 4-4　一个包含命令块的 if 语句

当 if 语句中包含命令块时，if 语句的条件为 true 时就会执行命令块中的语句。命令块执行之后，下面的代码按照正常的顺序执行。当 if 语句的条件为 false 时，就会跳过命令块执行下面的语句。如果图 4-4 中的 if 语句包含的条件表达式的值为 false，那么就只执行命令块后面的 document.writeln()语句。

当创建一个 if 语句时，记住在对 if 语句的条件求值之后，就会执行条件后面的第一条语句或者命令块。if 语句后面的命令或命令块是否执行决定于 if 语句的条件是 true 还是 false。

有时很容易忘记把所有语句包含在命令块内，这些语句在 if 语句的条件表达式为 true 时执行。例如，考虑如下代码：

```
if(exampleVar==true)
var conditionTrue="conditionistrue";
alert(conditionTrue);
```

代码表面看上去是正确的。实际上，当条件为 true 时，代码能够正确运行。但是，当条件为 false 时，因为跳过了 conditionTrue 变量的声明，警告对话框就会显示“ undefined ”。为了解决这个问题，需要把两条语句包含在一个命令块中，如下所示：

```
if(exampleVar==true){
var conditionTrue="conditionistrue";
alert(conditionTrue);
}
```

现在，如果条件为 false，两条语句就都会被跳过，因为它们包含在一个命令块中。

等号运算符仅仅是可以用在 if 语句中的几个关系运算符之一。可以使用在第 3 章中学到的任何一个关系运算符完成布尔关系运算。也可以把逻辑运算符与关系运算符组合使用。图 4-5 显示了一个使用了关系运算符和逻辑运算符的 if 语句示例。

接下来，将会创建本章一开始看到的 CartoonQuiz.html 文件。在这个程序中，用户通过选择用<INPUT>标签创建的单选按钮来回答问题。没有选择时单选按钮显示为一个小的空心圆；选择之后，它里面会有一个黑点。一个单选按钮通常包含在一组单选按钮之中，而且一次只能选择这组单选按钮中的一个。术语“ 单选按钮 ”来自汽车无线电，它包含一组按钮，每一个都设置为一个无线电台。每一次只能选择一个汽车无线电按钮，同样，每一次也只能在一组单选按钮中选择一个单选<INPUT>按钮。包含在同一组中的单选按钮必须具有相同的 NAME 属性。

在这个版本的测验程序中，每一个问题都是立即打分。会创建包含单选按钮的表单，然后使用一系列 if 语句对问题进行打分。首先，要创建 HTML 文档和表单段，然后把对问题打分的 JavaScript 代码加入其中。

创建 CartoonQuiz.html 文档和它的表单段：

1. 启动文本编辑器或 HTML 编辑器，并创建一个新文档。
2. 输入<HTML>和文档的<HEAD>部分。这个部分包含一个<SCRIPT>…</SCRIPT>

标签对。稍后会使用<SCRIPT>…</SCRIPT>标签对来创建给测验打分的代码。

```
var exampleVar1 = 5;
if (exampleVar1 != 3) // not equal
document.writeln("This line prints.");
if (exampleVar1 > 3) // greater than
document.writeln("This line prints.");
if (exampleVar1 < 3)    // less than
document.writeln("This line does not print.");
if (exampleVar1 >= 3)    // greater than or equal
document.writeln("This line prints.");
if (exampleVar1 <= 3)    // less than or equal
document.writeln("This line does not print.");
var exampleVar2 = false;
if (exampleVar1 > 3 && exampleVar2 == true) // logical 'and'
document.writeln("This line does not print.");
if (exampleVar1 == 5 || exampleVar2 == true) // logical 'or'
document.writeln("This line prints.");
```

图 4-5　使用关系运算符和逻辑运算符的 if 语句

```
<HTML>
<HEAD>
<TITLE>Cartoon Quiz</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
//ADD CODE HERE
// STOP HIDING FROM INCOMPATIBLE BROWSERS -- >
</SCRIPT>
</HEAD>
```

3. 输入如下代码，其中包含起始<BODY>标签、显示在测验顶部的文本以及创建单选按钮的起始<FORM>标签。

```
<BODY>
<H1>CartoonQuiz</H1><P>
Answer all of the questions on the quiz, then select the
Score button to grade the quiz
<FORM>
```

4. 接下来输入第一个问题。四个单选按钮代表答案。因为一个单选按钮组中每一个按钮要求具有相同的 NAME 属性，这四个单选按钮的名字都为“ question1 ”。而且每一个按

钮根据答案号分配了一个相应的值：a、b、c、d。对于每一个单选按钮来说，onClick 事件把按钮的值发送给一个单独的计算答案得分的函数。需要注意每一个按钮的值都是使用形如 this.value 的 this 引用发送给函数的。

```
<B>1.What is the name of Walt Disney's famous mouse?</B><P>
<INPUT TYPE=radio NAME=question1 VALUE="a"
onClick="scoreQuestion1(this.value)">Mighty Mouse<BR>
<INPUT TYPE=radio NAME=question1 VALUE="b"
onClick="scoreQuestion1(this.value)">Mickey Mouse<BR>
<INPUT TYPE=radio NAME=question1 VALUE="c"
onClick="scoreQuestion1(this.value)">Marty Mouse<BR>
<INPUT TYPE=radio NAME=question1 VALUE="d"
onClick="scoreQuestion1(this.value)">Melvin Mouse<P>
```

帮助：可以通过拷贝第一个问题的输入按钮代码来快速创建程序中问题 2 到问题 5。如果使用拷贝和粘贴来创建输入按钮，要确保修改了每一个输入按钮名称中问题编号和所调用的函数。

5. 输入第二个问题。

```
<B>2.The character Buzz Light year was featured in which
animated film?</B><P>
<INPUT TYPE=radio NAME=question2 VALUE="a"
onClick="scoreQuestion2(this.value)">Fantasia<BR>
<INPUT TYPE=radio NAME=question2 VALUE="b"
onClick="scoreQuestion2(this.value)">Hercules<BR>
<INPUT TYPE=radio NAME=question2 VALUE="c"
onClick="scoreQuestion2(this.value)">Toy Story<BR>
<INPUT TYPE=radio NAME=question2 VALUE="d"
onClick="scoreQuestion2(this.value)">Mulan<P>
```

6. 输入第三个问题。

```
<B>3.Pluto is a dog. What is Goofey?</B><P>
<INPUT TYPE=radio NAME=question3 VALUE="a"
onClick="scoreQuestion3(this.value)">A bear<BR>
<INPUT TYPE=radio NAME=question3 VALUE="b"
onClick="scoreQuestion3(this.value)">A mule<BR>
<INPUT TYPE=radio NAME=question3 VALUE="c"
onClick="scoreQuestion3(this.value)">A horse<BR>
<INPUT TYPE=radio NAME=question3 VALUE="d"
onClick="scoreQuestion3(this.value)">Also a dog<P>
```

7. 输入第四个问题。

```
<B>4.Who was always trying to eat Tweety Bird?</B><P>
<INPUT TYPE=radio NAME=question4 VALUE="a"
onClick="scoreQuestion4(this.value)">Porky Pig<BR>
<INPUT TYPE=radio NAME=question4 VALUE="b"
onClick="scoreQuestion4(this.value)">Yosemite Sam<BR>
<INPUT TYPE=radio NAME=question4 VALUE="c"
onClick="scoreQuestion4(this.value)">Sylvester<BR>
<INPUT TYPE=radio NAME=question4 VALUE="d"
onClick="scoreQuestion4(this.value)">
Foghorn Leghorn<P>
```

8. 输入第五个问题。

```
<B>5.What is Winnie the Pooh's favorite food?</B><P>
<INPUT TYPE=radio NAME=question5 VALUE="a"
onClick="scoreQuestion5(this.value)">Honey<BR>
<INPUTTYPE=radio NAME=question5VALUE="b"
onClick="scoreQuestion5(this.value)">Molasses<BR>
<INPUT TYPE=radio NAME=question5 VALUE="c"
onClick="scoreQuestion5(this.value)">Peanut Butter<BR>
<INPUT TYPE=radio NAME=question5 VALUE="d"
onClick="scoreQuestion5(this.value)">Yogurt<P>
```

9. 输入如下代码结束<FORM>、<BODY>和<HTML>标签。

```
</FORM>
</BODY>
</HTML>
```

接下来创建计算每个问题的得分的函数。函数包含评价答案的 if 语句。

1．使用如下的计算第一个问题得分的函数代替//ADD CODE HERE 行。如果用户的答案正确会显示“ Correct Answer ”。如果用户的答案不正确就会显示“ Incorrect Answer ”。

```
function scoreQuestion1(answer) {
        if (answer == "a")
                alert("Incorrect Answer");
        if (answer == "b")
                alert("Correct Answer");
        if (answer == "c")
```

```
        alert("Incorrect Answer");
    if (answer == "d")
        alert("Incorrect Answer");
}
```

2. 输入 scoreQuestion2()函数。

```
function scoreQuestion2(answer) {
    if (answer == "a")
        alert("Incorrect Answer");
    if (answer == "b")
        alert("Incorrect Answer");
    if (answer == "c")
        alert("Correct Answer");
    if (answer == "d")
        alert("Incorrect Answer");
}
```

3. 输入 scoreQuestion3()函数。

```
function scoreQuestion3(answer) {
    if (answer == "a")
        alert("Incorrect Answer");
    if (answer == "b")
        alert("Incorrect Answer");
    if (answer == "c")
        alert("Incorrect Answer");
    if (answer == "d")
        alert("Correct Answer");
}
```

4. 输入 scoreQuestion4()函数。

```
function scoreQuestion4(answer) {
    if (answer == "a")
        alert("Incorrect Answer");
    if (answer == "b")
        alert("Incorrect Answer");
    if (answer == "c")
        alert("Correct Answer");
    if (answer == "d")
        alert("Incorrect Answer");
```

```
}
```

5. 输入 scoreQuestion5()函数。

```
function scoreQuestion5(answer) {
    if (answer == "a")
        alert("Correct Answer");
    if (answer == "b")
        alert("Incorrect Answer");
    if (answer == "c")
        alert("Incorrect Answer");
    if (answer == "d")
        alert("Incorrect Answer");
}
```

6. 保存文件为盘上 Tutorial.04 目录下的 CartoonQuiz1.html。在 Web 浏览器中打开 CartoonQuiz1.html 文件。当选择了问题的答案后，立刻就会知道答案是否正确。图 4-6 显示了问题 1 选择一个错误答案时的输出。

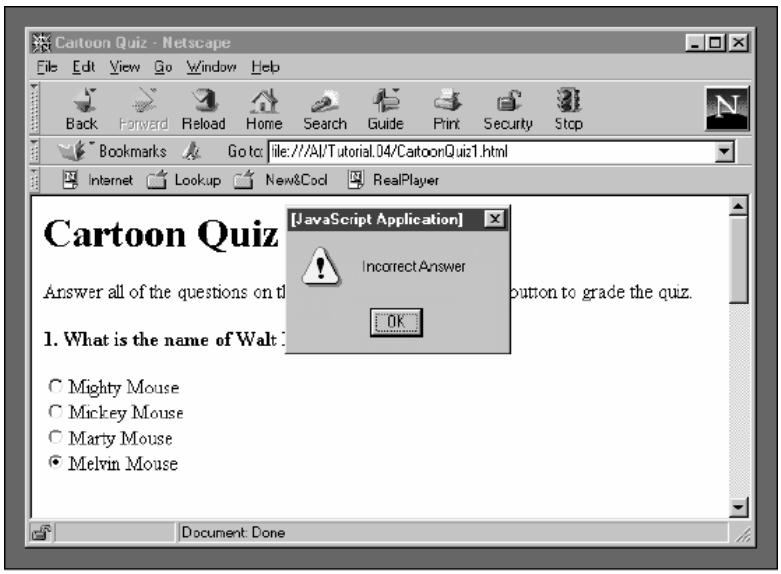


图 4-6　CartoonQuiz1.html 的输出

7. 关闭 Web 浏览器窗口。

4.1.2　if…else 语句

使用 if 语句时，也可以包含一个 else 分句，在 if 语句的条件表达式求值为 false 时运

行另外一段代码。例如有一个使用 if 语句的程序，其条件表达式对一个询问用户是否在股票市场投资的确认对话框的返回值求值。如果条件为 true（用户按下“ OK ”按钮），if 语句显示一个关于推荐股票的 Web 页面。如果条件为 false（用户按下“ Cancel ”按钮），则 else 分句中的语句会显示一个关于其他投资机会的 Web 页面。一个包含 else 分句的 if 语句称为 if…else 语句。可以把 else 分句看作是 if 语句的条件返回 false 值时的一个备用方案。if…else 语句的语法如下所示：

```
if (conditional expression ) {
statement(s);
else{
statement(s);
}
}
```

也可以使用命令块构造 if…else 语句，如下所示：

```
if (condition ) {
statements ;
}
else {
statements ;
}
```

提示：一个 if 语句可以不包含 else 分句。但是，else 语句只能在 if 语句中使用。

图 4-7 显示了一个 if…else 语句示例。

```
var today = "Tuesday"
if (today == "Monday")
        document.writeln("Today is Monday");
else
        document.writeln("Today is not Monday");
```

图 4-7　if…else 语句示例

在图 4-7 中，today 变量赋值为 Tuesday。因为 if (today == "Monday")的值为 false，程序控制传递给 else 分句，它执行 document.writeln("Today is not Monday");语句打印字符串“ Today is not Monday ”。如果 today 赋值为 Monday，则 if (today == "Monday")的值为 true，就会执行 document.writeln("Today is Monday");语句。只有一个语句组被执行：或者是 if 语句后的语句组或者是 else 分句后的语句组。任何一个语句组执行之后，if…else 后面的所有代码都会正常执行。

前面创建的 CartoonQuiz1.html 中的 JavaScript 代码使用了多个 if 语句计算测验结果。

尽管使用多个 if 语句能够完成所需功能，但使用 if…else 语句可以简化程序。接下来会使用 if…else 代替多个 if 语句来简化 CartoonQuiz1.html 程序。

为 CartoonQuiz1.html 添加 if…else 语句：

1. 回到 CartoonQuiz1.html 文件，并立即把它存为 CartoonQuiz2.html。

2. 因为测试正确的答案只需要 if 语句，可以把所有的不正确答案放在 else 分句中。使用一个 if…else 语句代替多个 if 语句修改每一个给问题打分的函数。下面的代码显示了修改后的 scoreQuestion1()函数是什么样子。

```
if (answer == "b")
   alert("Correct Answer");
else
   alert("Incorrect Answer");
```

3. 保存 CartoonQuiz2.html 文档并在 Web 浏览器中打开它。这个程序和只包含 if 语句的程序具有相同的功能。

4. 关闭 Web 浏览器窗口。

4.1.3　嵌套 if 和 if…else 语句

当使用类似于 if 或 if…else 语句的控制结构进行判断时，可能会希望控制结构执行的语句能够进行其他判断。假设有一个程序使用一个 if 语句询问用户是否喜欢运动。如果用户回答“是”，您可能希望运行另外一个 if 语句询问用户喜欢团队运动还是个人运动。因为可以在 if 语句或 else 分句中包含所需的任何代码，所以也可以包含其他 if 或 if…else 语句。一个包含在 if 或 if…else 语句中的 if 语句称为嵌套 if 语句。类似地，一个包含在 if 或 if…else 语句中的 if…else 语句称为嵌套 if…else 语句。可以使用嵌套 if 和 if…else 语句执行除原来的条件求值之外的条件求值。例如，下面的代码在执行 document.writeln()语句前执行两次条件求值。

```
var number = 7;
if (number > 5)
        if (number < 10)
            document.writeln(
                "The number is between 5 and 10.");
```

如果例子中的任何一个条件求值为 false，JavaScript 解释器都会跳过 if 语句的其他部分。

也可以使用嵌套 if…else 语句。在嵌套 if 语句中使用 else 分句时，需要注意 else 分句是与哪个 if 语句相关联的。一个 else 分句属于距离它最近的 if 语句。考虑如下代码：

```
var number = 7;
var numberRange;
if (number > 5)
        if (number > 10)
                numberRange = "The number is greater than 10.";
else
        numberRange = "The number is less than 5.";
document.writeln(numberRange);
```

因为一个 else 分句属于距离它最近的 if 语句，所以上面例子中的 else 分句属于第二个 if 语句。在此示例中，因为 number 变量不大于 10，所以 else 分句会被执行，它把“ The number is less than 5 ”赋给 numberRange 变量。这个结果是不正确的，因为第一个 if 语句已经判定 number 变量是大于 5 的。这时很容易错误地把 else 分句当作是第一个 if 语句的一部分，因为缩进是这样对齐的。记住，JavaScript 解释器并不能识别缩进或空格，使用它们只是让阅读更加容易。下面的代码是该程序的一个修改后的版本，它使用了另外一个 else 分句，使程序可以正确运行。

```
var number = 7;
var numberRange;
if (number > 5)
        if (number > 10)
                numberRange = "The number is greater than 10.";
        else
                numberRange = "The number is less than 10.";
else
        numberRange = "The number is less than 5.";
document.writeln(numberRange);
```

if 语句的嵌套没有层数限制。但是，if 语句嵌套过深会变得很难理解。在图 4-8 中，每一个 if 语句都对 country 变量进行比较，然后程序控制转移到包含在一个 else 分句中的下一个 if 语句。

注意最后一个 else 分句没有包含 if 语句。在一系列嵌套 if 语句中，最后一个不包含 if 语句的 else 语句通常用来在前面的 if 语句都与条件表达式不匹配时执行一项任务。

设计图 4-8 中的程序的更为有效的方法是把每一个 if 语句与前面的 else 分句放在同一行。记住 JavaScript 解释器并不能识别空格。因此语句 else if (country == "Germany")的功能和把它分成两个单独的行完全相同。图 4-9 显示了图 4-8 中程序的一个改进版本。

CartoonQuiz2.html 文件中的 JavaScript 代码的效率稍微有点低，因为它包含了多个执行本质上相同的测验打分任务的函数。接下来会修改 CartoonQuiz.html 文件中的 JavaScript 代码，在其中仅包含一个使用 if…else 语句检查所有问题的正确答案的函数。

```
var country = "France";
if (country == "Spain")
        document.writeln("Buenos Dias");
else
        if (country == "Germany")
                document.writeln("Guten Tag");
        else
                if (country == "Italy")
                        document.writeln("Buon Giorno");
                else
                        if (country == "France")
                                document.writeln("Bonjour");
                        else
                                document.writeln(
                                        "I don't speak your language!");
```

图 4-8　使用嵌套 if 语句的问候程序

```
var country = "France";
if (country == "Spain")
        document.writeln("Buenos Dias");
else if (country == "Germany")
        document.writeln("Guten Tag");
else if (country == "Italy")
        document.writeln("Buon Giorno");
else if (country == "France")
        document.writeln("Bonjour");
else
        document.writeln("I don't speak your language!");
```

图 4-9　改进的使用嵌套 if 语句的问候程序

为 CartoonQuiz 程序添加嵌套 if…else 语句：

1. 回到 CartoonQuiz2.html 文件，并立即把它存为 CartoonQuiz3.html。

2. 删除<SCRIPT>…</SCRIPT>标签对中的五个函数。

3. 输入这个检查所有答案的单一函数的第一行：function scoreQuestions(number, answer) {。您会发送给 scoreQuestions()函数一个 answer 参数，这和使用对每一个单独的问题打分的函数相同。同时要把一个新的参数 number 发送给 scoreQuestions()函数，它代表问题的编号。

4. 回车并输入起始 if 语句检查问题编号是不是 1。如果是的话，一个嵌套 if…else 语

句负责检查问题的答案。

```
if (number == 1) {
        if (answer =="b")
                alert("Correct Answer");
        else
                alert("Incorrect Answer");
}
```

5. 输入问题编号 2 使用的 if…else 语句。

```
else if (number == 2) {
        if (answer =="c")
                alert("Correct Answer");
        else
                alert("Incorrect Answer");
}
```

6. 输入问题编号 3 使用的 if…else 语句。

```
else if (number == 3) {
        if (answer =="d")
                alert("Correct Answer");
        else
                alert("Incorrect Answer");
}
```

7. 输入问题编号 4 使用的 if…else 语句。

```
else if (number == 4) {
        if (answer =="c")
                alert("Correct Answer");
        else
                alert("Incorrect Answer");
}
```

8. 输入问题编号 5 使用的 if…else 语句。

```
else if (number == 5) {
        if (answer =="a")
                alert("Correct Answer");
        else
```

```
            alert("Incorrect Answer");
}
```

9. 输入 scoreQuestions()函数的结束花括号（}）。

10. 在五个 <INPUT> 标签内把 onClick 事件处理器调用的函数修改为 scoreQuestions(number, this.value)，把 number 参数修改为正确的问题编号。例如，问题 1 的事件处理器应该是：scoreQuestions(1, this.value)。

11. 保存 HTML 文档并在 Web 浏览器中打开它。这个程序应该与包含多个 if 语句和多个函数的程序具有相同的功能。

12. 关闭 Web 浏览器窗口。

4.1.4　switch 语句

另外一个经常用来控制程序流程的 JavaScript 语句是 switch 语句。switch 语句控制程序流程的方法是根据一个表达式的值来执行一系列特定的语句。switch 语句在一个 switch 语句中把表达式的值和一个特殊的标签进行比较，然后执行与标签相关联的语句。例如，假设程序中有一个名为 favoriteMusic 的变量，switch 语句能够计算这个变量的值（这个变量就是一个表达式）并且与 switch 结构中的标签进行比较。switch 语句中可能包含几个标签，例如 Jazz、Rock 或 Gospel。如果 favoriteMusic 变量等于 Rock，那么 Rock 标签部分的语句就会执行。尽管可以使用 if 或者 if…else 语句完成相同的功能，但是 switch 语句可以更容易地组织不同的被执行的代码“分支”。

一个 switch 结构由以下部分组成：关键字 switch、一个表达式、一个起始花括号、一个 case 标签、关键字 break、一个 default 标签、可执行语句和一个结束花括号。switch 语句的语法如下所示：

```
switch   (expression ) {
        case     label:
                 statement(s);
                break;
        case     label :
                 statement(s);
                break;
       ...
        default :
                 statement(s);
}
```

一个 switch 语句中的标签称为 case 标签，它们标识特定的代码段。一个 case 标签由关键字 case 后跟一个文字值或变量名以及一个冒号组成。JavaScript 比较 switch 语句的表达

式返回的值和 case 关键字后面的文字值或变量值。如果能够找到一个匹配，就会执行这个 case 标签的语句。例如，case 标签 case 3.17：表示一个浮点数值 3.17，如果 switch 语句的表达式值等于 3.17，就会执行 case 3.17：标签的语句。在同一个 switch 语句中可以使用各种各样的数据类型作为 case 标签。图 4-10 显示了不同的 case 标签的示例。

提示：一个 case 标签可以后跟一条或者多条语句。但与 if 语句不同的是，一个 case 标签中的多条语句不需要封装在一个命令块内。

```
case exampleVar:                    // variable name
    statement(s)
case "text string":                 // string literal
    statement(s)
case 75:                            // integer literal
    statement(s)
case -273.4:                        // floating-point literal
    statement(s)
```

图 4-10　case 标签示例

提示：在其他程序语言中，例如 Java 和 C++，要求一个 switch 语句中所有的 case 标签具有相同的数据类型。

在 switch 语句中使用的另外一类标签是 default 标签。default 标签中包含 switch 语句中的条件表达式与任何一个 case 标签都不匹配时执行的语句。default 标签由关键字 default 后跟冒号组成。

执行一个 switch 语句时，条件表达式返回的值按照出现的顺序与每一个 case 标签进行比较。只要发现一个匹配的标签，就会执行它的语句。和 if…else 语句不同，在一个特定 case 标签的语句执行之后程序运行不会自动退出 switch 结构。与此相反，switch 语句继续对列表中的其他 case 标签求值。当发现一个匹配的 case 标签后，对其他 case 标签求值可能是不必要的。如果使用一个带有许多 case 标签的大型 switch 语句，对额外的 case 标签求值可能会使程序变慢。

在 switch 完成了所需任务之后立刻结束 switch 语句是一种非常好的程序设计方式。一个 switch 语句在 JavaScript 解释器遇到结束花括号（}）或者 break 语句时自动结束。break 语句通常用来退出 switch 语句和其他程序控制语句如 while、do…while、for 和 for…in 循环语句。在 switch 语句完成所需任务后，应该在每一个 case 标签中包含一个 break 语句。

提示：在第 2 节中会学到更多有关循环语句的知识。

图 4-11 显示了一个包含在函数中的 switch 语句的例子。调用函数时传递了一个 americanCity 变量，switch 语句比较 americanCity 参数的内容与 case 标签。如果找到一个匹配，会返回城市所在的州名并使用 break 语句结束 switch 语句。如果没有匹配，就使用 default

标签返回值“ United States ”。

```
function city_location(americanCity) {
        switch (americanCity) {
                case "Boston":
                        return "Massachusetts";
                        break;
                case "Chicago":
                        return "Illinois";
                        break;
                case "Los Angeles":
                        return "California";
                        break;
                case "Miami":
                        return "Florida";
                        break;
                case "New York":
                        return "New York";
                        break;
                default:
                        return "United States";
        }
}
document.writeln(city_location("Boston"));
```

图 4-11　包含一个 switch 语句的函数

可以使用 if 语句或者 switch 语句处理相同的流程控制过程。如果需要对一个单独的表达式求值，使用 switch 语句的效率更高。例如，回顾一下图 4-9 中使用一系列 if…else 语句创建的问候程序。这个程序能够工作，但却不是最有效率的，因为比较 country 变量和一个特定的国家名称的条件求值重复了好几次。可以使用 switch 语句写一个效率更高的相同程序，如图 4-12 所示。

接下来将修改 CartoonQuiz 程序，在 scoreAnswers()函数中使用 switch 语句代替嵌套 if…else 语句。改进的程序中的每一个 case 语句检查从函数的 number 参数传来的问题编号。Switch 语句使程序具有更好的感觉，因为它不需要像 if…else 结构那样多次检查问题编号。

为 CartoonQuiz 程序添加 switch 语句：

1. 回到 CartoonQuiz3.html 文件，并立即把它存为 CartoonQuiz4.html。
2. 把 scoreQuestions()函数中的 if…else 语句修改为如下的 switch 语句。

```
switch (number) {
   case 1:
```

```
var country = "Germany";
switch (country) {
        case "Spain":
                document.writeln("Buenos Dias");
                break;
        case "Germany":
                document.writeln("Guten Tag");
                break;
        case "Italy":
                document.writeln("Buon Giorno");
                break;
        case "France":
                document.writeln("Bonjour");
                break;
        default:
                document.writeln("I don't speak your language");
}
```

图 4-12　使用一个 switch 语句的问候程序

```
        if (answer =="b")
                alert("Correct Answer");
        else
                alert("Incorrect Answer");
        break;
case 2:
        if (answer =="c")
                alert("Correct Answer");
        else
                alert("Incorrect Answer");
        break;
case 3:
        if (answer =="d")
                alert("Correct Answer");
        else
                alert("Incorrect Answer");
        break;
case 4:
        if (answer =="c")
                alert("Correct Answer");
        else
                alert("Incorrect Answer");
        break;
case 5:
```

```
        if (answer =="a")
                alert("Correct Answer");
        else
                alert("Incorrect Answer");
        break;
}
```

3. 保存 HTML 文档并在 Web 浏览器中打开它。这个程序和使用嵌套 if…else 语句的程序具有相同的功能。

4. 关闭 Web 浏览器窗口。

4.1.5　总结

- ✧ 流程控制是决定一个程序中语句执行顺序的过程。
- ✧ if 语句用来在一个条件表达式返回 true 值时执行特定的程序代码。
- ✧ 命令块指的是包含在一对花括号中的多条语句，和包含在花括号中的函数语句类似。
- ✧ 对 if 语句的条件求值为 true 时会执行条件后面的第一条语句或者命令块。
- ✧ 不管 if 语句条件表达式的值是 true 还是 false，if 语句后面的命令或命令块都会执行。
- ✧ else 分句在 if 语句的条件表达式求值为 false 时运行另外一段代码。
- ✧ 在一个 if…else 结构中，只有一个语句组被执行：或者是 if 语句后的语句组或者是 else 分句后的语句组。任何一个语句组执行之后，if…else 后面的所有代码都会正常执行。
- ✧ 一个包含在 if 或 if…else 语句中的 if 语句称为嵌套 if 语句。类似地，一个包含在 if 或 if…else 语句中的 if…else 语句称为嵌套 if…else 语句。
- ✧ switch 语句控制程序流程的方法是根据一个表达式的值来执行一系列特定的语句。
- ✧ switch 语句中的 case 标签用来标识特定的代码段。
- ✧ default 标签中包含 switch 语句中的条件表达式与任何一个 case 标签都不匹配时执行的语句。default 标签由关键字 default 后跟冒号组成。
- ✧ 执行一个 switch 语句时，条件表达式返回的值按照出现的顺序与每一个 case 标签进行比较。只要发现一个匹配的标签，就会执行它的语句。
- ✧ break 语句通常用来退出 switch 语句。

4.1.6　问题

1. 决定一个程序中语句执行顺序的过程称为_____。

a. 过程处理

b. 流程控制
c. 程序布局
d. 结构构成

2. 下面哪一个 if 语句的语法是正确的？

a. if (myVariable == 10);
alert("Your variable is equal to 10. ");
b. if myVariable == 10
alert("Your variable is equal to 10. ");
c. if (myVariable == 10)
alert("Your variable is equal to 10. ");
d. if (myVariable == 10),
alert("Your variable is equal to 10. ");

3. 一条 if 语句可以包含多条语句，它们_____。

a. 在 if 语句的结束分号后执行
b. 不包含在一个命令块内
c. 不包含其他 if 语句
d. 包含在一个命令块内

4. 执行完一个 if 语句后会发生什么？

a. 执行紧随 if 语句的语句
b. 结束程序
c. if 语句继续循环
d. 重复 if 语句中的第一个匹配的 case 标签

5. 在 if 语句中可以使用哪些运算符？

a. 只有关系运算符
b. 只有逻辑运算符
c. 关系运算符和逻辑运算符
d. 在 if 语句中不能使用运算符

6. 下面使用的 else 分句哪一个语法是正确的？

a. else(document.write("Printed from an else clause. "));
b. else document.write(“Printed from an else clause. ");
c. else "document.write("Printed from an else clause. ") ";
d. else; document.write("Printed from an else clause. ");

7. 以下哪个选项是对的？

a. if 语句必须包含一个 else 分句
b. else 分句可以不用在 if 语句中
c. if 语句可以不包含 else 分句
d. else 分句不能用在 if 语句中

8. if 语句中可以嵌套多少 if 语句？

a. 0

b. 1

c. 5

d. 没有限制

9. switch 语句控制程序流程的方法是根据_____来执行一系列特定的语句。

a. if…else 语句的结果

b. 运行的 JavaScript 的版本

c. if 语句是否在一个函数内执行

d. 条件表达式返回的值

10. switch 语句中的 case 标签通常用来_____。

a. 标识特定的代码段

b. 对条件表达式求值

c. 指明要被 JavaScript 解释器忽略的代码

d. 为其他程序员留下注释

11. 以下哪个选项使用的不是 switch 语句中的 case 标签的正确语法？

a. case "text string"：

b. case 5.48 ：

c. case myVariable

d. case ：

12. 当一个 switch 语句的条件表达式返回的值与所有 case 标签都不匹配时，标签中的_____语句被执行。

a. exception

b. else

c. error

d. default

13. 可以使用语句_____ 退出 switch 语句。

a. break

b. end

c. quit

d. complete

4.1.7　练习

1. 创建一个 Web 页面，使用 if…else 语句提醒用户是否希望看到一个个性化的问候。如果用户选择“是”，显示一个提示对话框询问用户姓名，然后在一个警告对话框中显示输入的姓名。如果用户选择“否”，则在一个警告对话框中显示一个一般的问候信息。把

文档存为 PersonalGreeting.html。

2. 使用 switch 语句重写以下语句：

```
if (sport == "golf")
    alert("Golf is played on a golf course.");
else if (sport == "tennis")
    alert("Tennis is played on a tennis court.");
else if (sport == "baseball")
    alert("Baseball is played on a baseball diamond.");
else if (sport == "basketball")
    alert("Basketball is played on a basketball court.");
```

3. 使用 if…else 语句重写以下语句：

```
switch (writer) {
    case "Ernest Hemingway":
        alert(writer + " wrote Islands in the Stream");
        break;
    case "William Faulkner":
        alert(writer + " wrote The Sound and the Fury");
        break;
    case "Toni Morrison":
        alert(writer + " wrote Song of Solomon");
        break;
    case "F. Scott Fitzgerald":
        alert(writer + " wrote Tender is the Night");
        break;
    case "Henry Miller":
        alert(writer + " wrote Tropic of Capricorn");
        break;
}
```

4. 创建一个提示用户所居住的州名的程序。创建一个判断结构检查州名，并显示一个包含州所在地区的名称的警告对话框：North、South、East、West、Midwest、Southwest 等。把文件命名为 Regions.html。并思考对于此类程序应该使用的最好的控制结构是什么。

4.2 循　　环

本节目标

在本节中将会学习到如何使用：

- ✧ while 语句
- ✧ do …while 语句
- ✧ for 语句
- ✧ for…in 语句
- ✧ with 语句
- ✧ continue 语句

4.2.1　while 语句

目前学到的语句都是以一种线性的方式执行，一条接着一条。if、if…else 和 switch 语句选择一个代码分支执行，然后接着执行下面的语句。但是如果想重复执行一条相同的语句、函数或代码段时怎么办？例如想重复执行 5 遍、10 遍或者 100 遍。举个例子来说，可能想执行同样的计算，一直找到一个特定的数字。循环语句指在一个特定的条件为真时，或者直到一个特定的条件为真时止，重复地执行一条或一系列语句。

一个最简单的循环语句是 while 语句。while 语句用来在一个给定的条件表达式为真时重复地执行一条或一系列语句。下面是 while 语句的语法：

```
while(条件表达式){
    语句体；
}
```

与 if…else 和 switch 语句一样，while 语句需要测试的条件表达式包括在紧跟着关键字 while 的圆括号之中。当条件表达式的值为真时，则语句体中的语句或命令块将会重复执行。每次循环语句的重复称为一次迭代。当条件表达式的值变为假时，循环将会结束，程序转向 while 下面的语句执行。

一个 while 语句会一直重复执行直到它的条件表达式为假时止。当需要的任务已经执行完毕的时候，需要结束 while 语句。要结束 while 语句，必须在其中包括跟踪循环并且可以改变条件表达式的值的代码。可以用一个计数器来跟踪 while 语句的进程。计数器是一个可以随着循环语句的每次迭代增加或减少的变量。

提示：许多程序员通常将一个计数器变量命名为 count、counter，或者其他相似的名称。字母 I、J、K 和 L 也是常用作计数器的名称。用一个类似 count，或者字母 I（增加的计数器）或者其后的字母，可以帮助您（或其他程序员）记住这个变量用来作计数器。

下面的简单代码演示了使用 while 语句的一个例子。程序中声明了一个变量 count，并赋给初始值为 1。然后 count 变量用在 while 语句的条件表达式（count<=5）中。只要 count 变量小于或者等于 5，while 语句将会一直循环。在 while 语句体里，语句 document.writeln() 打印出 count 变量的值，然后 count 变量增加 1。while 语句会一直循环，直到 count 变量增加到 6 为止。

```
var count=1;
while(count<=5){
document.writeln(count);
++count;
}
document.writeln("You have printed 5 numbers. ");
```

这段代码的执行会打印出数字 1 至 5，表示了循环的每次迭代过程。当计数器到 6 后，循环结束时，消息“ You have printed 5 numbers ”被打印到显示界面。图 4-13 显示了输出结果。

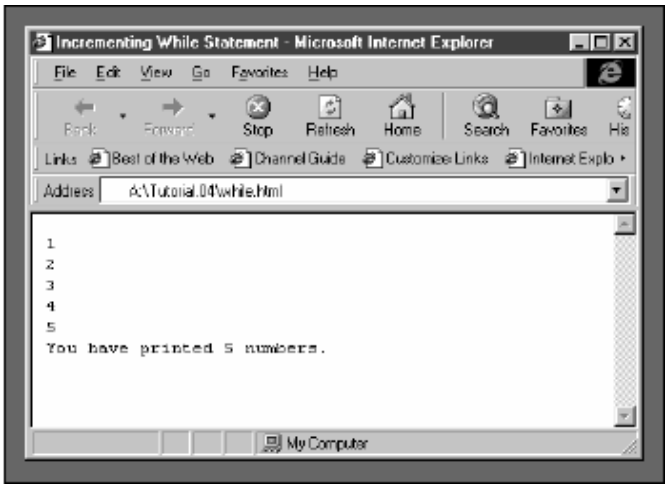


图 4-13　使用增 1 操作符的 while 语句的输出结果

在while循环里，还可以通过降低计数器变量的值来控制循环的重复执行。考虑下面的代码：

```
var count = 10;
while (count > 0) {
        document.writeln(count);
--count;
}
document.writeln("We have liftoff.");
```

在这个例子里，count变量的初值为10，用自减操作符（--），每次迭代将count的值减1。当count的值大于0时，while语句中的代码将count变量的值打印出来。当count的值等于0时，while循环停止执行，然后将后面的语句打印出来。程序的输出在图4-14中。

有很多方法来控制计数变量的值和用计数器控制 while 循环的重复过程。下面的例子用*=赋值操作符将 count 变量的值乘以 2。当 count 变量的值达到 128 后，while 语句结束。图 4-15 显示了这个程序的输出结果。

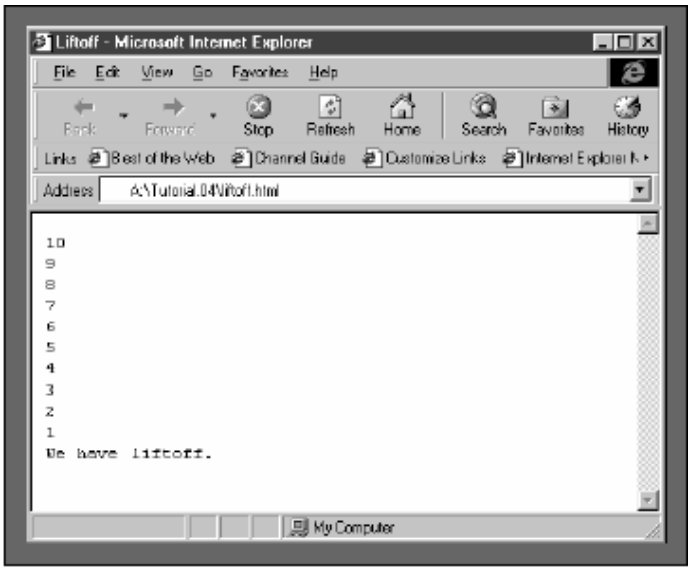


图 4-14　用减 1 操作符的 while 语句的输出结果

```
var count=1;
while (count<=100){
    document.writeln(count);
    count*=2;
}
```

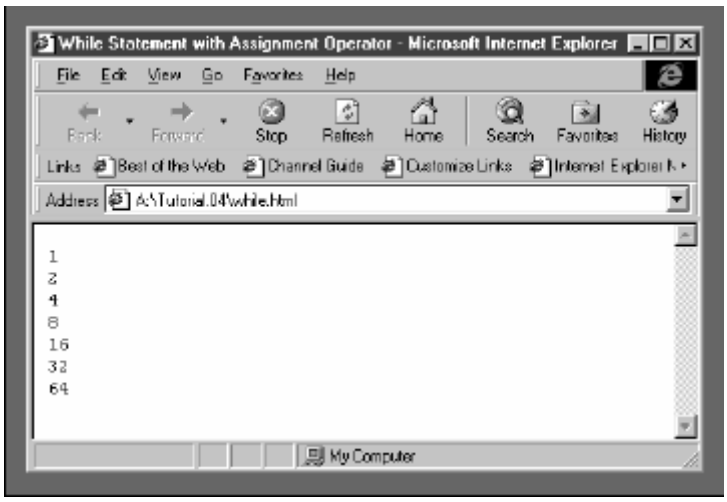


图 4-15　运用 * =赋值操作符的 while 语句的输出结果

在 while 语句中，包括监视条件表达式的代码是十分重要的。还必须在 while 语句中包括部分改变条件表达式值的代码。如果在 while 语句体中没有包括改变条件表达式值的代码，程序将陷入无限循环中。无限循环是指一个循环语句永远不会结束，因为它的条件表达式没有得到更新或者永远不会变为 false。考虑以下的 while 语句：

```
var count=1;
while(count<=10){
    alert("The number is "+ count);
}
```

在上面的例子里，虽然在 while 语句里包括了检查 count 变量取值的条件表达式，但是在 while 语句体里没有包含改变 count 变量值的代码。在循环的每个迭代过程中，count 变量将一直保持值为 1。在这种情况下，一个包含文本串“ The number is 1 ”的提示对话框会一直出现了再出现，不管按了多少次对话框上的“ OK ”按钮都不管用。

提示：在大多数情况下，必须强制一个陷入无限循环的 Web 浏览器结束。当然，强制结束一个应用程序的方法会随着计算机系统的不同而有所不同。对 Windows NT/95/98 操作系统来说，可以通过按下 Ctrl+Alt+Delete，调出任务列表或任务管理器，来强制结束一个应用程序。当结束程序对话框出现后，单击“ 结束任务 ”按钮就可以。

下面会创建一个演示使用 while 语句的程序。这个程序用 while 语句检查在提示对话框中输入的“ speed ”。然后 speed 被用来作为 while 语句的计数器。如果一直输入一个低于 65 的速度，会一直接收到一个要求您输入新速度的对话框。如果输入一个高于 65 的速度，或者输入一个等于或小于 0 的速度，while 循环就会结束。

创建一个限制速度的程序:

1．打开文本编辑器或 HTML 编辑器，创建一个新的文档。

2．输入开始的标记。

```
<HTML>
<HEAD>
<TITLE>Speed Limit</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

3．输入函数的开始部分，该函数有一个 speed 参数：function speedLimit(speed) {，然后按回车。

4． 添加一个 while 语句的开始部分，该循环语句在 speed 变量小于或等于 65 的情况下，一直循环：while(speed <= 65) {，然后按回车。

5．在 while 语句体中添加下面的语句，将一个提示对话框中的输入赋给名为 newSpeed 的变量。

```
var newSpeed = prompt("Your speed is " + speed
      + ". Please enter a new speed", "");
```

6．按回车，然后添加一个 if…else 语句，检查 speed 是否大于 65。如果 speed 大于 65，

弹出一个警告对话框，然后调用 break 语句。

```
if (newSpeed > 65) {
        alert("You are speeding!");
        break;
}
```

7．按回车，然后输入一个 if…else 语句，检查 speed 是否小于或等于 0。如果 speed 小于或等于 0，则弹出一个警告对话框，然后调用 break 语句。

```
else if (newSpeed <= 0) {
        alert("You are stopped!");
        break;
}
```

8. 然后在 while 语句后面添加最后的 else 子句。

```
else {
        speed = newSpeed;
}
```

9. 添加下面的关闭代码。

```
   }
}
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
```

10. 输入下面的语句行，输入<BODY>标记，创建一个<SCRIPT>部分，在该部分中调用 speedLimit()函数，该函数赋给一个初值 55。

```
<BODY>
<H1>Select Reload or Refresh to restart the
SpeedLimit program.</H1><P>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

```
speedLimit(55);
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
```

11. 添加下面的关闭代码。

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

12. 在文件夹 Tutorial.04 中，保存文件为 SpeedLimit.html。在 Web 浏览器中打开文件并测试程序。图 4-16 显示了程序的输出。

13. 关闭 Web 浏览器窗口。

图 4-16　SpeedLimit.html 的输出

4.2.2　do…while 语句

另一个类似于 while 语句的 JavaScript 循环语句是 do…while 语句。do…while 语句首先将一个语句或语句体执行一次。然后，如果条件表达式的值为真，就会一直重复执行循环。do…while 语句的语法如下所示：

```
do {
      语句或语句体；
}while (条件表达式)
```

就像在语法描述中看到的，语句体在条件表达式检查之前执行。不像简单的 while 语句，do…while 语句总会在条件表达式计算之前执行一次。

下面的 do…while 语句在条件表达式计算 count 变量的值之前执行了一次。因此，一行

文本“ The count is equal to 2 ”被打印出来。当条件表达式（count<2）执行时，do…while 语句结束，因为 count 变量的值等于 2，因此条件表达式的返回值为 false。

```
var count = 2;
do {
    document.writeln("The count is equal to "+ count);
    ++count;
} while (count<2);
```

需要注意的是，在这个 do…while 例子里，包括了一个计数器。和 while 语句一样，需要在语句体里包括可以改变部分条件表达式值的代码，以防止无限循环的发生。

在下面的例子里，while 语句将永远得不到执行，原因是 count 变量的值没有落在条件表达式的范围之内。

```
var count =2;
while(count >2){
     document.writeln("The count is equal to "+ count);
}
```

图 4-17 演示了一个用数组打印一周的天数的 do…while 语句的例子。

```
<PRE>
<SCRIPT>
var daysOfWeek = new Array();
daysOfWeek [0 ] = "Monday";
daysOfWeek [1 ] = "Tuesday";
daysOfWeek [2 ] = "Wednesday";
daysOfWeek [3 ] = "Thursday";
daysOfWeek [4 ] = "Friday";
daysOfWeek [5 ] = "Saturday";
daysOfWeek [6 ] = "Sunday";
var count = 0;
do {
document.writeln(daysOfWeek [count ]);
++count;
} while (count < 7);
</SCRIPT>
</PRE>
```

图 4-17　do…while 语句的例子

在图 4-17 的例子里，创建了一个包含了一周的天数的数组。声明了一个 count 变量并初始化为 0。注意，数组的下标或索引是从 0 开始的。因此，在这个例子里，语句 daysOfWeek[0] 代表 Monday，星期一。do…while 语句的第一次迭代打印出“ Monday ”，然后 count 变量每次增 1。在 while 语句中的条件表达式检查 count 变量是否小于 7。只要 count 变量的值比 7（7 是比数组 daysOfWeek 的下标上限大的数字）小，循环将继续。图 4-18 显示了这个程序在 Web 浏览器上的输出。

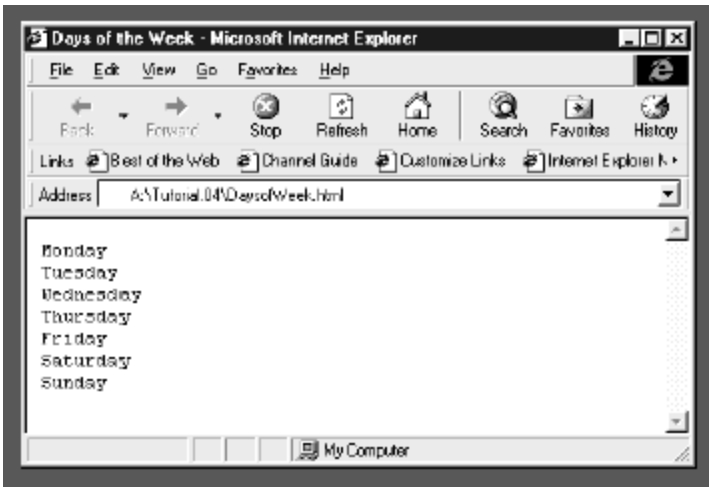


图 4-18　一周天数程序在 Web 浏览器的输出

下面会学着修改限速程序，用 do…while 语句代替 while 语句。用 do…while 语句就消除了给 speedLimit()函数的 speed 参数赋初值的必要。speed 变量是在 do 语句的第一次循环时初始化的，然后 while 语句检查它的值，看是否需要继续循环。

用 do…while 语句的限速程序：

1. 返回 speedLimit.html 文件，然后另存为 SpeedLimit2.html。
2. 删除在 speedLimit()函数中的 speed 参数。
3. 修改 while 语句，创建一个如下的 do…while 语句。注意，不再需要 newSpeed 变量。还要注意，最后的 else 子句也不需要了。

```
do {
        var speed = prompt("Your speed is " + speed
                + ". Please enter a new speed", "");
        if (speed > 65) {
                alert("You are speeding!");
                break;
        }
        else if (speed <= 0) {
                alert("You are stopped!");
                break;
```

```
        }
    } while(speed <= 65)
```

4. 将 speedLimit()函数调用中的 55 删除。

5. 保存文件，然后在 Web 浏览器中打开。当第一次打开时，会在对话框中显示“ You speed is undefined ”。记住，当一个变量被声明后，它包含一个未定义的初值，直到给它赋一个值为止。因为将 speedLimit()函数的 speed 参数删除掉了，speed 变量包含了一个未定义的初值，直到在对话框中输入一个限速取值。

6. 关闭 Web 浏览器窗口。

4.2.3　for 语句

也可以在代码中使用 for 语句来实现循环。for 语句，在一个给定的条件表达式为真时，用来重复执行一条或一系列语句。for 语句执行的基本功能和 while 语句相似。如果在 for 语句中的构建器为真，则执行 for 语句，并且会在条件表达式为真时继续重复执行，直到条件表达式为假时才停止执行。在 for 语句和 while 语句之间一个基本的不同点是：除了条件表达式外，还可以在 for 语句的构建器中包括初始化计数器和在每次迭代中改变计数器取值的代码。for 语句的语法结构如下所示：

```
for (initialization expression; condition; update statement) {
    statement(s);
     }
```

当 JavaScript 解释器遇到一个 for 循环，按照以下步骤执行：

1. 首先启动初始化表达式。例如，如果在一个 for 循环的初始化表达式是 var count=1;，那么变量 count 被声明，并被赋初值为 1。当执行到一个 for 语句时，初始化表达式只启动一遍。

2. 计算 for 循环的条件表达式。

3. 如果在第 2 步中计算的条件表达式为真，则执行 for 循环里的语句，转到步骤 4 中，然后处理过程再从步骤 2 开始。如果步骤 2 中的条件表达式返回为假，for 循环结束，继续执行紧跟在 for 语句后的语句。

4. 执行 for 语句构建器中的更新语句。例如，变量 count 可能每循环一次就增加 1。

可以任意忽略在 for 语句构建器中的三个部分，但是必须用逗号将各部分分开。如果忽略了其中的一个部分，那么必须保证在 for 循环体中的代码能够结束循环，以免程序进入无限循环。

图 4-19 显示了一个利用 for 语句打印数组的值的例子。

就像在例子中看到的，在构建器中包括了对计数器的初始化、计算值和增加三个部分。没有必要在 for 语句前声明 count 变量，也没有必要在 for 语句体中增加 count 变量的值。

图 4-20 显示了一个快餐输出程序。

```
var fastFoods = new Array();
fastFoods [0 ] = "pizza";
fastFoods [1 ] = "burgers";
fastFoods [2 ] = "french fries";
fastFoods [3 ] = "tacos";
fastFoods [4 ] = "fried chicken";
for (var count = 0; count < 5; ++count) {
document.writeln(fastFoods [count ]);
}
```

图 4-19　利用 for 语句打印数组的值

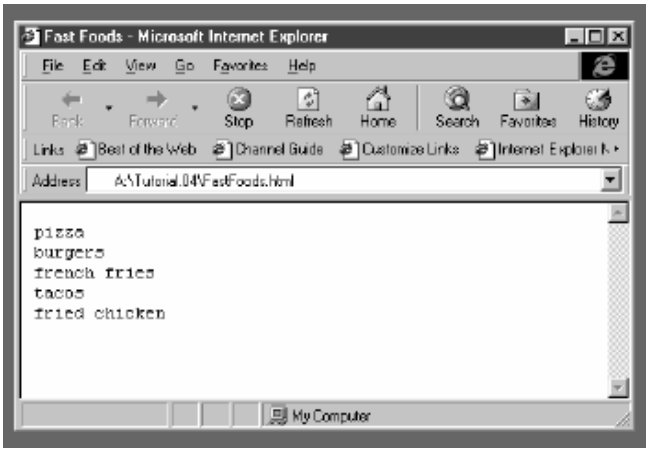


图 4-20　快餐程序的输出

可以用 for 语句创建比 while 语句更有效率的循环语句。原因在于用 for 语句时，没有必要写过多的代码。考虑下面的 while 语句：

```
var count=1;
while(count <=5){
   document.writeln(count);
        ++count;
}
```

上面的 while 语句可以用一个更有效率的 for 语句代替：

```
for(var count=1; count<=5; ++count){
```

```
    document.writeln(count);
}
```

但是在一些情况下，用 while 语句比 for 语句更合适。如果不需要用计数器更新条件表达式，或者计数器必须在循环体中更新，则 while 语句更合适。下面的代码依赖一个从确认对话框返回的布尔值，而不是计数器来控制程序。

```
var I=true;
while(I==true)
    I=confirm(
        "Do you want to redisplay this dialog box?");
```

也可以用 for 语句来完成同样的任务，但是在上述的例子中，在 for 语句的构建器中，用来更新计数器的第三部分就没有必要了。如果用 for 语句代替上面例子中的 while 语句，必须把更新代码放在构建器之外，这样就创建了一个无限的循环。注意，必须保留构建器中的逗号。下面的代码执行了和上例同样的功能，但是由于在构建器里总是把变量 I 的值置为真，以执行一个无限的循环。在确认对话框里按下“取消”按钮，可以置 I 的值为假，但是不管按“取消”按钮多少次，在构建器中总会把 I 再置为真，这样一直无限循环下去。

```
for(var I=true; I==true; I=true){
            I=confirm(
"Do you want to redisplay this dialog box?");
}
```

为了避免上例的无限循环，必须把构建器中的更新部分去掉，如下所示：

```
for(var I=true; I==true; ){
            I=confirm(
"Do you want to redisplay this dialog box?");
}
```

图 4-21 显示了一个和图 4-18 一样“周天”的例子。但是在这个例子中，用 for 语句代替了 do…while 语句。注意 count 变量的声明，条件表达式和 count 变量的增加，现在都包含在构建器里了。用 for 语句代替了 do…while 语句，这样就没必要编写这么多行的代码，可以使编程变得简单一些。

下面将创建一个卡通的测验程序的最终版本，该程序用嵌套了一个 if 语句的 for 语句来计算测验的分数。如果创建的 for 语句比 if、if…else 和 switch 语句复杂，那它就可以减少相当的代码行数。还需要一个“得分”按钮，以便用户在做完测试后可以结束整个测验，而不必用一个答案一个答案地来看得分。

创建一个卡通的测验程序的最终版本：

```
<PRE>
<SCRIPT>
var daysOfWeek = new Array();
daysOfWeek [0 ] = "Monday";
daysOfWeek [1 ] = "Tuesday";
daysOfWeek [2 ] = "Wednesday";
daysOfWeek [3 ] = "Thursday";
daysOfWeek [4 ] = "Friday";
daysOfWeek [5 ] = "Saturday";
daysOfWeek [6 ] = "Sunday";
for (var count = 0; count < 7; ++count) {
document.writeln(daysOfWeek [count ]);
</SCRIPT>
</PRE>
```

图 4-21 一个 for 语句的例子

1. 在磁盘中的 Tutorial.04 的文件夹中，打开 CartoonQuiz4.html 文件，并把它存为 CartoonQuizFinal.thml。

2. 将在<HEAD>部分中的整个 scoreQuestion()函数全部删除掉，然后增加下面的两行代码来创建两个数组：answers[]和 correctAnswers[]。数组 answers[]保存了测试的每个答案，数组 correctAnswers[]保存每个问题的正确答案。在代码中将正确的答案赋值给 correctAnswers[]的每个元素。

```
var answer = new Array(5);
var correctAnswers = new Array(5);
correctAnswers[0]="b";
correctAnswers[0]="c";
correctAnswers[0]="d";
correctAnswers[0]="c";
correctAnswers[0]="a";
```

3. 按下回车键并输入下面的函数，这个函数将用户的答案赋给对应的 answers[]的元素。程序将实际的问题编号（1~5）送给该函数，函数的调用是通过用每个选择按钮的 onClick 事件来完成的。要想将答案传递给正确数组元素，必须从变量 question 中将 1 提取出来，因为数组的元素是从 0 开始的。

```
Function recordAnswer(question, answer){
    answers[question -1] = answer;
}
```

4. 输入为测试记分的函数的开始部分：function scoreQuiz(){。可以从一个新的“得分”按钮来调用该函数。

5. 回车，然后输入 var totalCorrect = 0;用它声明一个变量并把它初始化为 0。变量 totalCorrect 记录了正确答案的个数。

6. 回车，然后输入计算测试的 for 循环的开始部分：for(var count = 0; count <5; ++count){。将一个命名为 count 的计数器初始化为 0，因为 0 是数组的起始下标。条件表达式用来检查 count 变量是否超出了数组的范围。最后，循环的每次迭代都将 count 变量增加 1。

7. 回车，然后在循环体中添加下面的 if 语句。该语句将 correctAnswers[]中的正确答案和 answers[]中的答案进行比较。如果两个答案匹配，变量 totalCorrect 就会增加 1。

```
if (answers[count] == correctAnswers[count])
    ++totalCorrect;
```

8. 输入 for 循环的结束花括号。然后添加一个提示对话框的代码，用来显示用户答对了多少题目。

```
}
alert("You scroed " + totalCorrect
   + "out of   5   answers correctly! ");
```

9. 添加 scoreQuiz()函数的结束花括号}。

10. 将选择按钮的 onClick 事件处理代码中调用的函数 scoreQuestions()改为 recordAnswer()。

11. 最后，在第五个问题的最后一个选择按钮后面添加下面的<INPUT>标记。用该 INPUT 标记创建一个命令按钮，该按钮的 onClick 事件的代码中调用了 scoreQuiz()函数。

```
<INPUT TYPE=button VALUE="Score"onClick =
"scoreQuiz();"><P>
```

12. 保存文件，然后在 Web 浏览器窗口中打开该文件。回答所有的五个问题，然后按下“得分”按钮。在 Web 浏览器窗口中显示的结果会和图 4-22 相似，具体的得分取决于答对了多少道题。

4.2.4　for…in 语句

在第 2 章里，学会了用对象和属性来工作。正如回忆起来的，用一种特殊的构建函数来创建对象。包含在一个构建函数中的变量，属于该构建函数创建的对象的数据，通常我

们称之为属性。for…in 语句就是用来对一个对象的所有属性循环执行语句或命令块的语句。该语句相当有用，例如，如果想打印出一个对象的所有属性的名称。for…in 语句的语法如下所示：

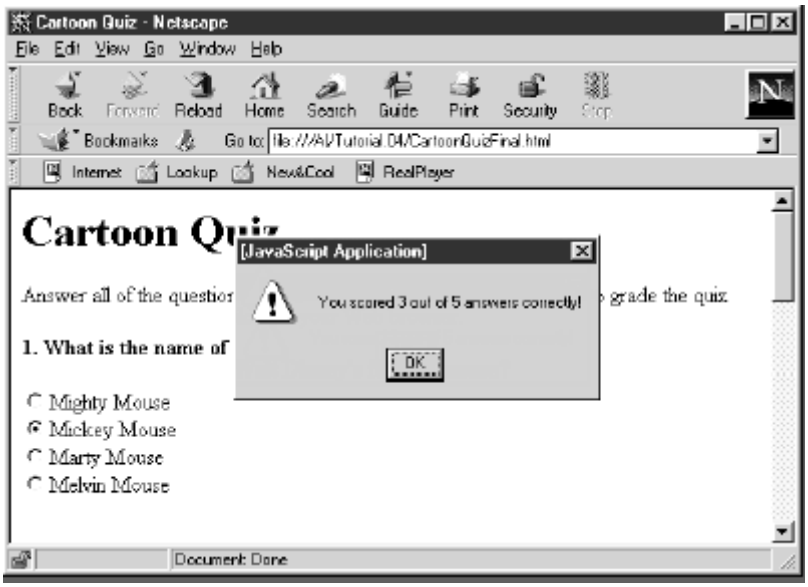


图 4-22 CartoonQuizFinal.html 的输出

```
for (variable in object){
   statement(s);
}
```

在 for…in 语句的构建器中，变量的名字代表了一个独立的对象。在构建器中的对象名称 object 代表了程序中已经声明的一个对象。不像其他的循环语句，for…in 语句不用任何的计数器或者代码操作循环的进行。相反地，for…in 语句自动将对象 object 加在循环体中的属性变量之前，执行与该属性变量有关的语句，然后转移到对象的下一个属性继续开始执行。当到达该对象的最后一个属性时，for…in 语句会自动结束。对 for…in 语句的一个典型的应用是检索一个对象的所有属性，正如图 4-23 所示。

在图 4-23 中的 for…in 语句中，变量 prop 把握着对象 lovestock 的每个属性，而对象 lovestock 在构建函数 Animal 中声明。然后语句 document.writeln()把每个属性的名称显示到 Web 浏览器的窗口上，如下所示：

```
animal_type
animal_sound
animal_transport_mode
```

提示：在 JavaScript 中，并没有方法控制一个对象里的属性是如何分配给 for…in 语句里的变量的。

```
function Animal(type, sound, transport_mode) {
        this.animal_type = type; // object property
        this.animal_sound = sound; // object property
        this.animal_transport_mode = transport_mode;
                // object property
livestock = new Animal("cow", "moo", "walk");
              // instantiate object
for (prop in livestock) {        // this for loop prints
        document.writeln(prop); // the names of the properties
}
```

图 4-23　用 for…in 语句打印一个对象的所有属性

用 for…in 语句的其中一个好处是它的枚举功能，或者说可以给一个对象的每个属性指定索引，这种情况类似于对一个线性的数组进行索引。可以用一个带有下标的对象属性来存取包含在属性中的值。在图 4-24 中的代码和图 4-23 中的代码很相似，惟一不同的地方是，在图 4-23 中的 document.writeln() 语句已经被替换成了 document.writeln (livestock[prop]);。

在图 4-24 代码中的每次循环打印出的是每个属性的取值（“ cow ”、“ moo ”和“ walk ”），而不是属性的名称。在代码中，将带有 prop 下标的 lovestock 对象传递给 document.writeln() 方法。也可以用同样的方法打印数组的内容。然而，在 for…in 语句之外，就不能像索引数组的元素那样存取一个对象的枚举属性了；否则的话会产生错误。即语句 document. writeln(livestock[prop])如果在 for…in 循环体外的话，会引起错误。

```
function Animal(type, sound, transport_mode) {
        this.animal_type = type;
        this.animal_sound = sound;
        this.animal_transport_mode = transport_mode;
livestock = new Animal("cow", "moo", "walk");
for (prop in livestock) {
        document.writeln(livestock [prop ]);
}
```

图 4-24　利用 for…in 语句打印一个对象中的所有属性的名称

下面，创建一个用 for…in 语句打印一个对象的属性的程序：

1. 打开文本编辑器或 HTML 编辑器来创建一个新的文档。
2. 输入<HTML>和<HEAD>部分的起始标志。

```
<HTML>
```

```
<HEAD>
<TITLE>Car Properties</TITLE>
```

3. 输入<SCRIPT>部分的起始标志：

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!--HIDE FROM INCOMPATIBLE BROWSES
```

4. 输入下面的类构建器，该构建器创建一个 Car 的构建函数。

```
function Car(make, model, color, doors){
this.car_make =make;
this.car_model=model;
this.car_color=color;
this.car_doors=doors;
}
```

5. 输入下面的结束标志。

```
//STOP HIDING FROM INCOMPATIBLE BROWSERS
</SCRIPT>
</HEAD>
```

6. 添加下面的代码，开始 HTML 文档的主体；并且创建一个预格式化的文本容器。

```
<BODY>
<PRE>
```

7．添加 JavaScript 部分的起始语句，该部分调用<HEAD>部分中的函数。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!--HIDE FROM INCOMPATIBLE BROWSERS>
```

8. 输入 sports_car=new Car();在 Car 类的构建器函数基础上声明一个新的 sports_car 对象。

9. 输入下面的语句，给 sports_car 的每个属性指定取值。

```
sport_car.car_make="Triumph";
sports_car.car_model="Spitfire";
sports_car.car_color="Yellow";
sports_car.car_doors=2;
```

10. 按回车，然后输入下面的 for…in 语句，用于打印 sports_car 对象的属性。

```
for(prop in sports_car){
   document.writeln(sports_car[prop]);
}
```

11. 添加下面的结束标志：

```
//STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

12. 将文件保存到 Tutorial.04 文件夹中，命名为 SportsCar.html。在 Web 浏览器中打开 SportsCar.html 文件。图 4-25 显示了输出情况。

13. 关闭 Web 浏览器的窗口。

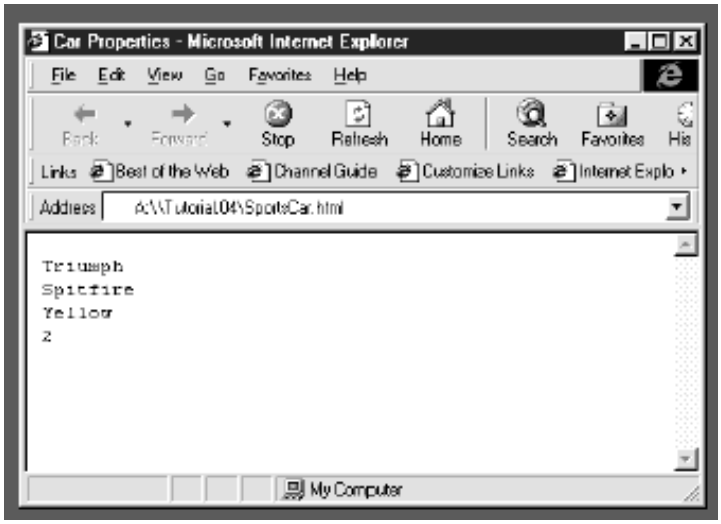


图 4-25　SportsCar.html 的输出

4.2.5　with 语句

另一个用来处理对象属性的语句是 with 语句。with 语句可以用在一系列语句中，消除重复输入对象名称的必要。如果将语句放在 with 语句的范围中，则就没有必要在每个属性前面重复输入对象的名称。可以在语句中简单地直接使用属性的名字，而不用在属性前面加上所属对象的名字。with 语句的语法如下所示：

```
with (对象) {
    语句（语句体）;
}
```

圆括号中是您想引用的对象的名字。花括号中，则是需要用到这个对象的属性的语句或语句体。下面的代码将 document.writeln()方法重复了 3 次：

```
document.writeln("Mark Twain wrote: ");
document.writeln("Everybody talks about the weather, ");
document.writeln("but nobody does anything about it.");
```

在上面的代码里，完全可以去掉每个语句前面的 document 对象，代之以 with 语句，新的代码如下所示：

```
with (document) {
writeln("Mark Twain wrote: ");
writeln("Everybody talks about the weather, ");
writeln("but nobody does anything about it.");
}
```

通常，with 语句用来简化重复输入对象名称的繁琐。图 4-26 显示了一个利用 with 语句的程序，该程序赋值给一个对象的各属性，这个对象具有不寻常的过长名字。

```
Function Animal(type, sound, transport_mode) {
this.animal_type=type;
this.animal_sound=sound;
this.animal_transport_mode =transport_mode;
}
animal_that_lives_in_the_forest = new Animal();
with (animal_that_lives_in_the_forest){
animal_type ="wolf";
animal_sound="growl";
animal_transport_mode="run";
}
```

图 4-26　利用 with 语句给对象的各属性赋值

如果在图 4-26 的程序里，不使用 with 语句的话，则给该对象的各个属性赋值的语句将不得不按照下面的方式来写：

```
animal_that_lives_in_the_forest.animal_type ="wolf";
animal_that_lives_in_the_forest.animal_sound="growl";
animal_that_lives_in_the_forest.animal_transport_mode="run";
```

接下来添加一个 with 语句到 SportsCar.html 文件里：

1. 打开 SportsCar.html 文件并将其保存为 SportsCar2.html。
2. 定位到下面的对 sports_car 对象属性赋值的语句。

```
sports_car.car_make="Triumph";
sports_car.car_model="Spitfire";
sports_car.car_color="Yellow";
sports_car.car_doors=2;
```

3. 利用 with 语句简化的代码如下。

```
with (sports_car){
car_make="Triumph";
car_model="Spitfire";
car_color="Yellow";
car_doors=2;
}
```

4. 保存文件，并在您的 Web 浏览器上重新打开。文件的输出应当如图 4-25 所示。
5. 关闭 Web 浏览器的窗口。

4.2.6　continue 语句

在上一节里，学习了利用 break 语句退出类似于 switch、while、do…while、for 和 for…in 语句的方法。break 语句终止 switch 或者循环语句的执行，转去执行其结构体后的语句。另一个用来继续循环的相似语句是 continue 语句。continue 语句终止一个循环语句的当前迭代，进而进行下一次迭代过程。当需要终止循环的当前迭代过程，进行下一个新的迭代过程时，可以使用 continue 语句。举个例子来说，也许会用一个包含有 for 语句的程序来处理一个存储有货物列表的数组。当货物的价值超过了 10 美元，需要在屏幕上显示出购买价格，存货等信息。但是，对于价值少于 10 美元的货物，就可以利用 continue 语句将其跳过并转到下一个迭代过程。图 4-27 的程序显示了一个包含有 break 语句的 for 循环。图 4-28 的程序显示了同样的循环，但是包含一个 continue 语句。

在图 4-27 和图 4-28 的程序里，都包含了一个 if 语句，用来检查 count 变量的值是否等于 3。在图 4-27 的程序里，如果 count 的值为 3，则 break 语句立即结束 for 循环。图 4-27 程序的输出如下：

1
2

```
for(var count = 1; count <=5; ++count) {
        if(count == 3)
                break;
        document.writeln(count);
}
```

图 4-27　包含有 break 语句的 for 循环

```
for(var count = 1; count <=5; ++count) {
        if(count == 3)
                continue;
        document.writeln(count);
}
```

图 4-28　包含有 continue 语句的 for 循环

在图 4-28 的程序里，当 count 变量等于 3 时，continue 语句同样结束当前的迭代过程，程序跳过打印数值 3。但是循环过程继续进行，直到条件表达式 count<=5 的值为假。图 4-28 的程序的输出如下：

1
2
4
5

下面将学习添加 continue 语句到 SportsCar.html 文件中，用其跳过打印 car_model 属性。

添加 continue 语句到 SportsCar.html 文件，跳过打印 car_model 属性：

1. 将 SportsCar.html 另存为 SportsCar3.html 文件。
2. 在“ document.writeln(sports_car[prop]); ”前面添加下面的代码。

```
if (prop=="car_model")
   continue;
```

当 for 循环执行到 car_model 属性时，经过 if 语句的判断执行 continue 语句，跳过对 car_model 属性的打印，执行下一个迭代过程。

3. 保存 SportsCar3.html 文件，在 Web 浏览器中将其打开。程序的输出将如图 4-29 所示。
4. 关闭 Web 浏览器窗口。

图 4-29　SportsCar3.html 文件的输出

4.2.7　总结

- ✧ 一个循环语句是用来重复执行一条语句或一段语句体的，前提条件是循环的条件表达式为真或直到指定的条件表达式为真时止。
- ✧ while 语句是用来重复执行一条语句或一段语句体的，前提条件是 while 语句中的给定的条件表达式值为真。
- ✧ 循环语句的每一次重复的执行过程称为一次迭代。
- ✧ 在 while 语句中，必须包含可以改变条件表达式的取值的代码，以便当完成需要的任务后结束循环。
- ✧ 计数器是循环语句中的一个变量，它随着循环的每一次迭代过程增加 1。
- ✧ 如果一个计数器变量超过了 while 语句的条件表达式的取值范围，则 while 语句将被跳过，得不到执行。
- ✧ 在一个无限循环里，由于条件表达式永远得不到更新，循环语句将永远不会结束。
- ✧ do…while 语句首先执行一次其中的语句或语句体，然后，只要给定的条件表达式为真，则重复执行。
- ✧ for 语句是用来重复执行一条语句或一段语句体的，前提条件是给定的条件表达式取值为真。
- ✧ 在 for 语句中，可以将构建器中的三部分都省略，但必须把逗号写上，以区分每个部分。如果省略了构建器中的一个部分，那么必须确定在循环体中包含了可以结束循环的语句，否则程序就有可能陷入无限循环中。
- ✧ for…in 语句用于对一个对象的所有属性执行同样的语句或命令块。
- ✧ 在 for…in 语句的构建器中标识的变量名称代表了一个单独的对象，表示在语句体中的处理默认的为该对象的属性。在 for…in 语句中，和其他循环不同的是，不需要计数器或类似功能的代码来控制循环。
- ✧ for…in 语句对一个对象的所有属性指定一个索引。

- ✧ 在一段语句体中，如果需要引用同一个对象的多个属性，可以用 with 语句消除重复输入对象名称的麻烦。
- ✧ continue 语句用来暂停一个循环语句的当前迭代过程，重新启动新的迭代过程。

4.2.8 问题

1. 循环语句的每一次重复的执行过程称为：

a. recurrence
b. iteration
c. duplication
d. re-execution

2. 计数器变量：

a. 用来标记循环语句重复执行的次数
b. 用来标记 web 浏览器被打开和关闭的次数
c. 用来标记一个 JavaScript 程序已经执行的次数
d. 仅仅被用在 if 或 if...else 语句中

3. 下面的哪一个 while 语句的语法是正确的？

a.
```
while(i<=5,++i){
        document.writeln(i);
  }
```
b.
```
while(i<=5){
        document.writeln(i);
        ++i;
    }
```
c.
```
while(i<=5);
        document.writeln(i);
        ++i;
```
d.
```
while(i<=5;document.writeln(i)){
        ++i
  }
```

4. 计数器变量：

a. 只能被增加
b. 只能被减少
c. 可以用任意的条件表达式改变
d. 不发生变化

5. 一个无限循环产生的原因是：

a. 忽略了决策机构的结束括号

b. 当一个条件表达式永远没法取值为假时

c. 当一个条件表达式永远没法取值为真时

d. 当执行一个 while 语句时

6. 在大多数情况下，如果 JavaScript 程序陷入了一个无限循环，需要采取什么措施？

a. 在 JavaScript 程序中加入一个 break 语句

b. 在 Web 浏览器中将 HTML 文档重新载入

c. 什么也不用做，程序会自动结束

d. 将 Web 浏览器强制关闭

7. 如果一个 do...while 语句的条件表达式取值为假，则 do...while 语句会执行多少次？

a. 不执行

b. 一次

c. 两次

d. 一直重复执行，陷入无限循环

8. 下面哪个 do...while 语句的语法是正确的？

a.
```
do while (i < 10) {
 alert("Printed from a do...while loop.");
  }
```
b.
```
do { while (i < 10)
      alert("Printed from a do...while loop.");
  }
```
c.
```
do {
      alert("Printed from a do...while loop.");
      while (i < 10)
  }
```
d.
```
do {
      alert("Printed from a do...while loop.");
  } while (i < 10);
```

9. 下面哪个 for 语句的语法是正确的？

a.
```
for (var i = 0; i < 10; ++i)
          alert("Printed from a for statement.");
```
b.
```
for (var i = 0, i < 10, ++i)
          alert("Printed from a for statement.");
```
c.
```
for {
      alert("Printed from a for statement.");
  } while   (var i = 0; i < 10; ++i)
```
d.
```
for (var i = 0; i < 10);
      alert("Printed from a for statement.");
```

```
++i;
```

10. 一个 for 语句的初始化表达式何时执行？
a. 当 for 语句开始执行时
b. 每次迭代过程执行一次
c. 当计数器变量增加时执行
d. 当 for 语句结束时执行
11. 在 for...in 语句中，应当使用哪种类型的计数变量？
a. 布尔变量
b. 一个可以被增加或减少的变量
c. 一个数组的长度属性
d. 在 for...in 语句里，没有必要使用计数器变量
12. 在 for...in 语句里，属性变量标识了：
a. 一个对象名称
b. 一个对象的属性的名称
c. 一个计数器变量
d. 一个类构建器的名称
13. 一个 for...in 循环何时结束？
a. 当关闭了 JavaScript 程序后，循环结束
b. 当按下 Ctrl+Break，循环结束
c. 当它的条件表达式取值为假时
d. 当对象的最后一个属性被处理完时
14. 用_____语句可以消除在一系列处理同一个对象的属性时，重复输入对象名称的繁琐操作。
a. having
b. include
c. with
d. contains
15. 用_____语句可以中止循环的当前迭代过程，不结束循环，而是转向新的迭代。
a. proceed
b. reiterate
c. restart
d. continue

4.2.9 练习

1. 使用for语句替代下面的while语句。

```
var count = 25;
while (count >= 0) {
        document.writeln("The current number is " + count);
        --count;
    }
```

2. 用 while 语句替代下面的 for 语句。

```
for(var i=1; i <= 15; ++i) {
        if (i == 10)
                break;
        else
                document.writeln("The current number is " + i);
}
```

3. 用 with 语句简化下面的程序。

```
function Employee(name, number, position, department) {
        this.employee_name = name;
        this.employee_social_security = number;
        this.employee_position = position;
        this.employee_department = department;
new_entry_level_employee = new Employee();
new_entry_level_employee.employee_name = "John Doe";
new_entry_level_employee.employee_social_security = "000-12-3456";
new_entry_level_employee.employee_position = "Intern";
new_entry_level_employee.employee_department = "Accounting";
document.writeln(new_entry_level_employee.employee_name);
document.writeln(new_entry_level_employee.employee_social_security);
document.writeln(new_entry_level_employee.employee_position);
document.writeln(new_entry_level_employee.employee_department);
```

4. 改写在练习 3 中创建的程序，用 for...in 语句打印 emloyee 的各属性。

5. 创建一个问询程序，该程序询问访问您网站的客户的一些个人信息，例如姓名、地址、职业等。将各信息存入一个数组，然后用 for 循环将所有信息打印到屏幕上。请将 HTML 文档命名为 Questonnarie.html。

第 5 章　窗口和帧

案例

WebAdventure 正在为一个大的公共动物园创建 Web 站点。动物园方面希望通过站点可以吸引更多的游客，扩大动物园募捐的影响。站点的一部分将是一个为孩子设计的虚拟的动物园，包括了动物图片、教育信息和游戏。您现在的任务是创建一个 HTML 文档，当用户单击动物名称的链接时，就将该动物的图片显示出来。在简单的 Web 页面里，当用户单击了一个链接后，一个新的 Web 页面将替代原来的页面。但是，现在您不想让动物的图片完全替代虚拟动物园原来的页面。解决的方法是创建一个带有帧的 HTML 文档，该文档可以包含独立的、在 Web 浏览器的窗口中可以调整的帧，其中每个帧都有自己独立的 HTML 文档和 image 文件。

预览虚拟动物园程序

在本章里，将创建一个“ 虚拟动物园 ”的 Web 页面，该页面显示用户单击的动物的一幅图片。动物名称列在其中的一个帧中，动物的图片显示在另一个帧中。

1．在光盘里有一个名为 Tutorial.05 的文件夹，在 Web 浏览器中打开其中的 Tutorial5_VirtualZoo.html 文件。图 5-1 显示了该程序在 Web 浏览器中的一个示例。

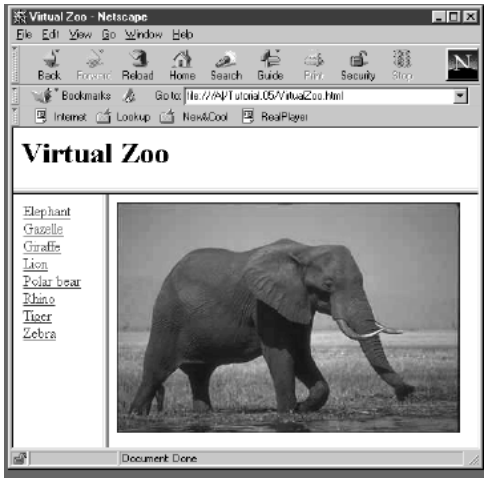


图 5-1　Tutorial5_VirtualZoo.html

2．在左边的帧中单击不同的动物的名称，则每个动物的图片会显示在右边的帧里。

3．如果预览完成，将 Web 浏览器的窗口关闭。

4．然后，在文本编辑器或 HTML 编辑器中打开 Tutorial5_VirtualZoo.html 文件，来检查它的代码。注意标签<FRAMESET>和<FRAME>。这些标签创建了组成该窗口的三个窗格。每个窗格都有独立 URL，该 URL 在<FRAME>标签的 SRC 属性里标识出来。

5．如果查看完了代码，则关闭文本或 HTML 编辑器。

5.1　用窗口工作

本节目标

在本节里将会学习：

- ✧ 关于 JavaScript 对象模型的知识
- ✧ 关于窗口对象的知识
- ✧ 如何打开和关闭窗口
- ✧ 如何用超时限定和时间间隔工作

5.1.1　JavaScript 对象模型

在某些情况下，可能需要利用 JavaScript 对 Web 浏览器进行控制。例如，可能需要改变正在显示的 Web 页面，或者需要往 Web 浏览器的状态条上写一些信息，或者需要控制 HTML 文档的某些元素。要想控制 Web 浏览器的窗口或 HTML 文档，需要利用 JavaScript 的对象模型。JavaScript 对象模型是 JavaScript 对象的一个体系，其中的每个对象都提供对 HTML 页面或 Web 浏览器的窗口的程序控制的功能。当然，每个对象处理的是不同的方面。可以利用 JavaScript 对象模型中对象的属性和方法对窗口帧和在 Web 浏览器中显示的 HTML 元素进行操作。在 JavaScript 对象模型中，不必专门创建任何对象或数组。当在 Web 浏览器中打开一个 HTML 页面时，所有对象已经被自动创建。JavaScript 对象模型显示在图 5-2 中。

提示：JavaScript 对象模型也被称为浏览器对象模型，客户端对象模型或者导航器对象模型。

提示：将会在第 2 节学习帧的有关知识。

从图 5-2 中，可以看到，窗口对象在 JavaScript 对象模型的最顶层。窗口对象指一个 Web 浏览器的窗口或其中的一个单独帧。Web 浏览器自动创建窗口对象，可以利用窗口对象的属性和方法控制 Web 浏览器的窗口。

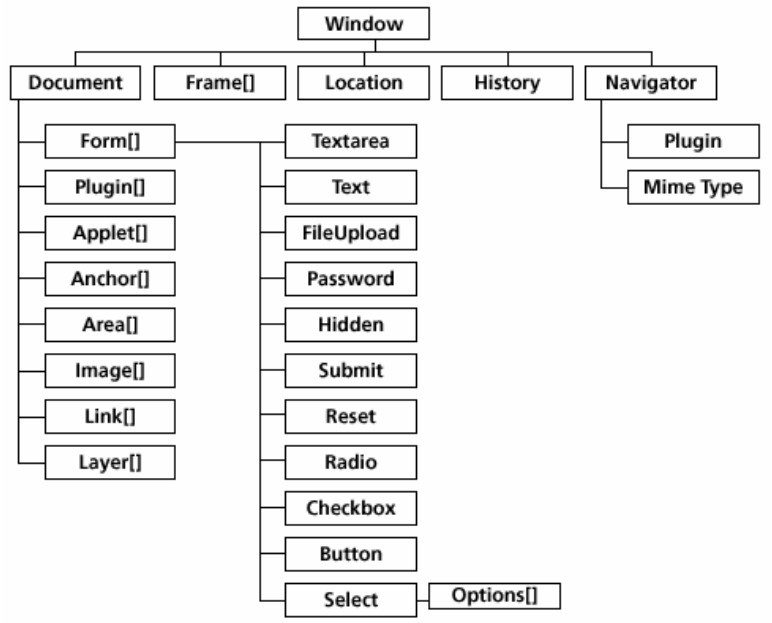


图 5-2 JavaScript 对象模型

在 JavaScript 对象模型中的另一个重要的对象是文档对象，它代表了显示在窗口中的 HTML 文档。文档对象在窗口对象的下一层。write()和 writeln()就是文档对象的方法。语句 document.write("This is an example");会在 Web 浏览器中的 HMTL 文档中添加文本“ This is an example ”。文档对象包括了存在于一个 HTML 文档中的所有 HTML 元素，并且每个元素在 JavaScript 中都有自己的对象。所以，文档对象是在一个 HTML 页面中创建的所有元素的父亲或祖先对象。例如表单对象，在 JavaScript 中代表了用<FORM>…</FORM>标签对创建的表单，就是从文档对象传下来，而文档对象又是从窗口对象传下来。再如复选按钮，在 JavaScript 里表示用<INPUT>标签创建的复选按钮，又是从表单对象传下来的。

提示：将会在第 7 章里学习到关于文档对象的更多知识。

在本书里，如果指 JavaScript 对象模型中的对象，首字母都会大写（Document 对象）。当在代码中使用对象时，请采用小写，因为实际上指的是对象的一个属性。例如，在语句 document.writeln("Text String")中，小写首字母的 document 实际代表了文档对象的 document 属性。

在图 5-2 列出的对象中，有一些代表了包含有其他对象的数组。在本图中，这样的对象都跟着中括号，例如 Form[]或者 Image[]对象。这些数组对象的元素则是 HTML 文档的元素创建而来。例如，图片对象就包含了一个图片的数组，该数组包括在 HTML 文档中所有的<IMG>标签标识出的图片。

在 JavaScript 的代码中引用对象时，必须把对象的所有祖先像属性层次一样都表示出

来。例如，考虑一个命名为 myImage 的<IMG>标签，如果需要引用 myImage 对象的 SRC 属性，必须包括 myImage 对象的祖先——文档对象。正确的语法应该是 document.myImage.src。考虑下面的代码，它创建了一个包括一个单行文本输入域的表单。

```
<FORM NAME="myForm">
<INPUT TYPE="text" NAME="myTextBox">
</FORM>
```

如果要在提示对话框中显示文本输入域的值，必须这样写代码：alert(document.myForm.myTextBox.value);对象 myForm 作为一个属性添加到 document 对象中，同时，myTextBox 对象也作为一个属性添加到 myForm 对象中。最终，文本对象的 value 属性返回了在文本输入域中的文本。如果用类似于 alert(myForm.myTextBox.value)或者 alert(myTextBox.value)的语句，将会产生错误，因为它们并未标明文本框对象的所有祖先。

当我们提一个对象的祖先时，并不需要把窗口对象包括在内。Web 浏览器自动地认为所做的操作都是针对当前显示的窗口的，同时，窗口对象也是 JavaScript 对象模型最顶层的对象。当然，可以把窗口对象作为一个对象的最高祖先列出来。例如，语句 alert(window.document.myForm.myTextBox.value); 和语句 alert(document.myForm.myTextBox.value)的效果是相同的。但是，既然 Web 浏览器自动地认为操作是对当前窗口而言的，所以也就没必要把窗口对象列出来。

提示：在 IE 里，通常没有必要把一个对象的所有祖先都列出来。例如，在上面的例子中，可以省略文档对象，用语句 alert（myForm.myTextBox.value）；同样能够正确工作。但是，如果想让自己的包含 JavaScript 代码的 HTML 文档在 IE 和 Navigator 中都工作的很好，最好将对象的所有祖先都包括进来。

提示：在某些情况下，必须包含窗口对象，通常这些情况是必须明确区分窗口对象和文档对象的时候。例如，事件处理代码自动地认为指的是文档对象而不是窗口对象。所以，在事件处理代码中，如果明确指定操作是针对 Web 浏览器窗口的话，必须包含窗口对象。

由于 Web 浏览器默认操作的是当前的窗口对象，所以在使用窗口的属性和方法时也没必要指明是针对窗口对象的。例如，弹出一个提示对话框的方法 alert()是窗口对象的一个方法。完整的语法应该是 window.alert(text);。虽然，在本书里看到的都是 alert(text);这样的语法，它省略了窗口对象，但也很好用。如果必须包括窗口对象，那么代码可能会变得非常长。例如，如果需要在取文本输入域的值的时候包括窗口对象，则代码就会是：

```
window.alert(window.document.myForm.myTextBox.value);
```

如果省略了对窗口对象的应用，就会缩短代码的长度，如下所示：

```
alert(document. myForm.myTextBox.value);
```

另一个引用窗口对象的方法是利用 self 属性，self 属性指当前的窗口对象。利用 Self 属性和用窗口对象的窗口属性有相同的作用。例如，下面的代码的作用是相同的：

```
window.alert("text string");
    self.alert("text string");
```

一些 JavaScript 的程序员喜欢用 window 属性，而另一些喜欢用 self 属性。究竟用哪种属性取决于您。但是，您在读其他程序员的代码时，必须明白，这两者都是指当前的窗口对象。

5.1.2 窗口对象

窗口对象包括了若干 Web 浏览器窗口信息的属性。举例来说，status 属性包含了显示于 Web 浏览器状态条中的信息。窗口对象还包含了操作 Web 浏览器窗口的各种方法。已经用过的窗口对象的方法包括 alert()、confirm()、prompt()，它们用来显示对话框。表 5-1 列出了窗口对象的常见属性，表 5-2 列出了窗口对象的常用方法。

表 5-1 窗口对象的属性

属　性	描　述
defaultStatus	写入状态条的默认文本
document	文档对象的引用
frames[]	窗口中的所有帧对象的数组
history	历史对象的引用
location	定位对象的引用
opener	打开另一个窗口的窗口对象
parent	包含当前帧的父帧
self	窗口对象的自我引用——与 window 属性相同
status	写入状态条的临时文本
top	包含当前帧的最高窗口对象
window	窗口对象的自我引用——与 self 属性相同
name	窗口的名称

表 5-2 窗口对象的方法

方　法	描　述
alert()	显示一个有一个 OK 按钮的简单消息对话框
blur()	将焦点从一个窗口移走
clearTimeout()	取消超时设定

续表

方　法	描　述
close()	关闭一个窗口
confirm()	显示一个有 OK 和 Cancel 按钮的确认对话框
focus()	将窗口置为当前活动窗口
open()	打开一个新窗口
prompt()	显示一个对话框提示用户输入信息
setTimeout()	在一段指定时间后执行一个函数

提示：Navigator 和 IE 都有自己独有的窗口对象的属性和方法。在本书中，只描述在 Navigator 和 IE 中都有的方法和属性。

5.1.3　打开和关闭窗口

Navigator 和 IE 都允许在原有的窗口之外，打开一个新的或已经存在的窗口。可能有几个原因需要打开一个新的 Web 浏览器的窗口。比如可能需要在一个单独的窗口中打开一个新页面，同时允许用户继续在原窗口浏览原来的页面。或者需要在一个另外的窗口显示诸如图片或订单等信息。

只要一个 Web 浏览器窗口被打开，一个代表当前窗口的新的窗口对象就被创建。只要在系统允许范围内，可以打开任意多的 Web 浏览器窗口，每个窗口显示不同的页面。例如，可以在一个窗口中显示微软的站点，在另一个窗口中显示网景的站点。在 Navigator 里，可以通过 “ 文件菜单 ” 中的 “ 新窗口 ” 命令手工打开一个新的 Web 浏览器窗口。在 IE 里，可以通过 “ 文件菜单 ” 中的 “ 新建 ” 子菜单中的窗口命令打开一个新窗口。当手工打开一个新的 Web 浏览器窗口后，新窗口打开显示和原窗口相同的 URL 或文件。例如，如果一个 Web 浏览器窗口显示 Course Technology 站点，则打开的新的窗口同样显示 Course Technology 站点。

用 JavaScript 语言，可以用窗口对象的 open()方法打开一个新的 Web 浏览器窗口。该方法的语法是：window.open("URL", "name",options)，参数 URL 表示要被打开的 Web 地址或文件名称。参数 name 用来赋值给新窗口对象的名称属性。注意要用引号将两个参数引起来。参数 options 代表了一个对新窗口的显示进行控制的字符串。可以在 open 方法里包括所有的三个参数，或者一个都不写。语句 window.open ("http://www.course.com");将在一个新的 Web 浏览器窗口中打开 Course Techonology 的站点，就像图 5-3 所示。如果省略 URL 参数，则会打开一个空白的网页。例如，语句 window.open();在 Web 浏览器中的显示如图 5-4 所示。

可以用窗口对象的名称属性来区别目标窗口，在该窗口里打开一个超文本的链接或帧。图 5-5 的程序中，建立了一个新的、命名为 targetWindow 的、空的 Web 浏览器窗口。原来的窗口包括了一个在新窗口 targetWindow 中打开 Course Technology 站点的链接。需要注意

的是，open()方法的URL参数是一个空串，因为我们要在新窗口中显示一个空页面。当open()方法执行完成后，语句 self.focus()将焦点返回到原 Web 浏览器窗口。

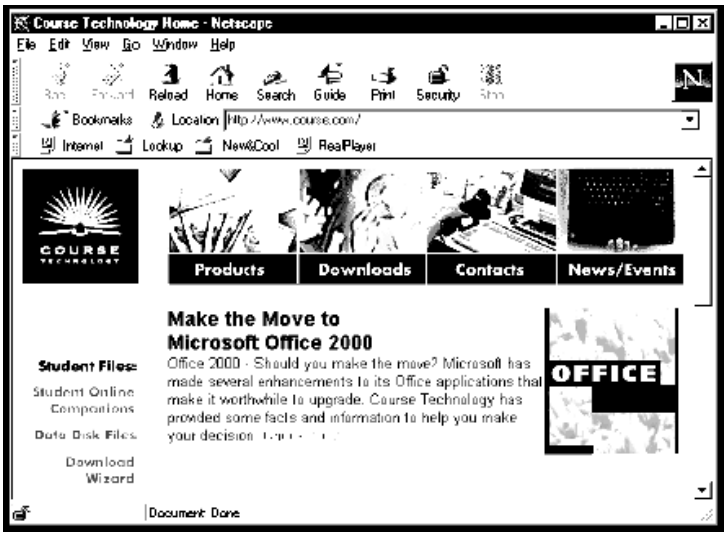


图 5-3　用带 URL 参数的 open()方法打开的 Web 浏览器窗口

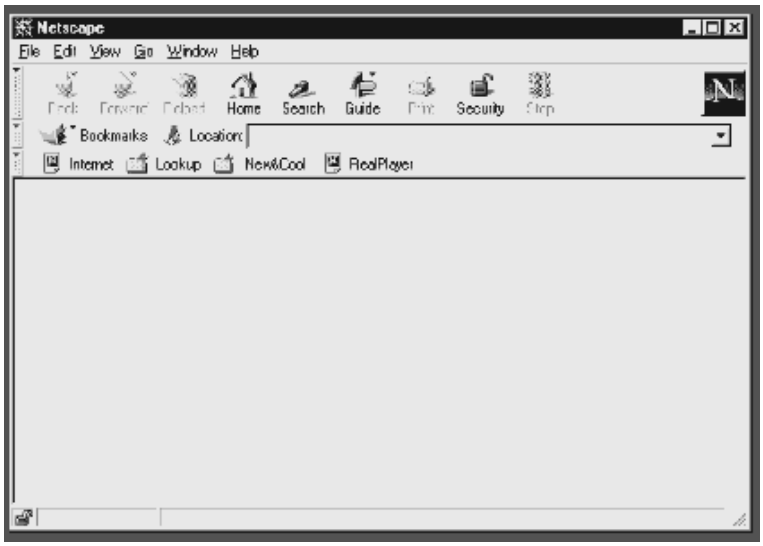


图 5-4　用 window.open()语句打开的 Web 浏览器窗口

提示：如果 open()方法的 name 参数已经被另一个存在的 Web 浏览器窗口占用，则 JavaScript 不会筹建新的窗口，而是将焦点转到已经存在的 Web 浏览器窗口中。

当打开一个新的窗口后，可以利用 open()方法的 options 参数来定制它的显示方式。表 5-3 列出了在 open()方法中常用的 options 参数的选项。

在表 5-3 的所有选项中，除了用来控制宽度和高度的选项外，其余的选项的取值不是

为 yes（真）就是为 no（假），通常用 1 代表 yes，用 0 代表 no。例如，显示时是否包含状态条，options 参数的字符串应当为 “ status=yes ”。高度和宽度的选项用代表像素值的整型数设置。举例来说，筹建一个高度为 200 像素，宽度为 300 像素的窗口，应当使用 “ height=200,width=300 ”。如果在 options 选项里包含多项参数，必须用逗号将其隔开。

```
<HEAD>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--HIDE FROM INCOMPATIBLE BROWSERS
window.open("", "targetWindow");
self.focus();
//STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
<A HREF=http://www.course.com TARGET="targetWindow">
Visit the Course Technology home page.</A>
```

图 5-5　一个包含了名称和目标参数使用的 JavaScript 程序

表 5-3　open()方法常见 options 选项

名　　称	描　　述
directories	包括目录按钮
height	设置窗口的高度
location	包括 URL 文本框
menubar	包括菜单条
resizable	设置新窗口是否可以调整大小
scrollbars	包括滚动条
status	包括状态条
toolbar	包括标准的工具条
width	设置窗口的宽度

如果忽略 options 参数，则新窗口将采用常用设置。但是，如果包括了 options 参数字符串，则必须将所有想对新窗口进行定制的项目都包含进去。也就是说，新窗口只会按照选定的参数进行设置，其余的参数将被忽略。下面的 open()方法创建了一个图 5-6 中的窗口。该窗口没有任何和用户交互的元素，例如菜单条、工具条、滚动条或其他和用户交互的窗口元素，原因是只在 options 参数里指定了高度和宽度属性。

```
window.open("http://www.course.com","Course",
"height=300,width=600");
```

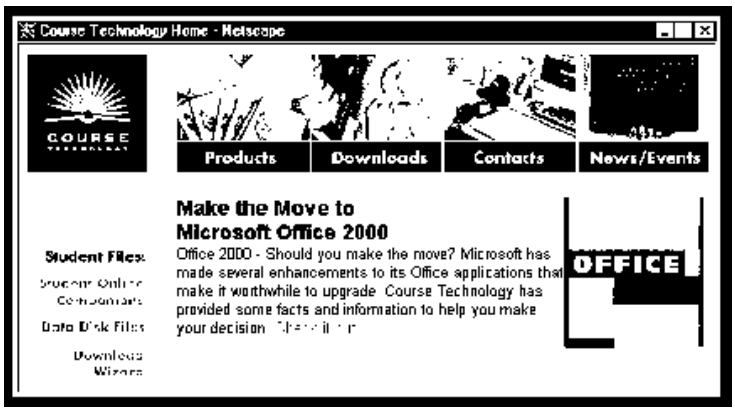


图 5-6　一个没有任何交互元素的 Web 浏览器窗口

比较下面的代码。下面的 open()方法就打开一个包括工具条和滚动条的窗口，显示结果如图 5-7 所示。

```
window.open("http://www.course.com","Course",
"height=300,width=600,toolbar=yes,scrollbars=yes");
```

图 5-7　一个有工具条和滚动条的窗口

提示：如果在 options 参数的字符串中包含了空格，有可能这些选项在 Navigator 中没法显示。因为 open()方法在 Navigator 和 IE 中都能正确工作，但是要排除在 options 参数字符串中包含空格的情况。

窗口对象的名称属性可以被用来标识一个带有超文本链接的目标窗口或表单，但是不能在 JavaScript 代码里面使用。如果需要在代码中控制新创建的窗口，必须把用 open()方法新创建的窗口对象赋值给一个变量。语句应为：

```
var newWindow=open("http://www.course.com");
```

该语句将表示新窗口的对象赋值给名为 newWindow 的变量。然后，就可以通过 newWindow 变量使用窗口对象的所有属性和方法。举例来说，如果想用窗口对象的 focus 方法将焦点转移到新窗口来，可以用语句 newWindow.focus();。用的最多的窗口对象方法是 close()，用语句 newWindow.close();，可以关闭 newWindow 变量表示的窗口。

提示：当需要列出一个对象的所有祖先时，没有必要将窗口对象包括在内。但是，文档对象同样包含名为 open()和 close()的方法，它们用来打开和关闭 HTML 文档。因此，在用窗口对象的 open()和 close()方法时，通常将前面的窗口对象表示出来，以便和文档对象的方法区分。

下面将尝试着创建一个程序，该程序在单独的窗口里显示一个包含北极熊照片的 HTML 文档。首先要做的是，建立一个可以打开一个新窗口的文档。

创建原文档：

1．打开文本编辑器或 HTML 编辑器创建一个新文档。

2．在文档中输入开头的若干标签。

```
<HTML>
<HEAD>
<TITLE>See the Polar Bear</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

3．创建下面的打开 PolarBear.html 文件的函数。包含 PolarBear.html 的新窗口将没有任何和用户交互的元素，仅仅是一个简单的“图片窗口”。在窗口对象的 open()方法中，options 字符串只包含高度和宽度参数。

```
function openPolarBear(){
      window.open("PolarBear.html", "PolarBear",
          "height=350,width=320");
}
```

4．输入下面的结束标签。

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
```

5．输入开始 HTML 文档正文的<BODY>标签。

6．输入<H1>See the Polar Bear</H1>在 HTML 文档中创建一个标题 1 的标签。

7．然后加入下面的代码，创建一个打开 PolarBear.html 文档的超级链接。该链接用标题 2 来显示。需要注意的是，程序并不在当前的窗口中打开 PolarBear.html 文档。可以看到，在 onClick 事件中，调用了 openPloarBear()函数，该函数用 window.open()函数在一个新窗口中打开文档。然后 onClick 事件返回一个为假的值，如果不返回这个为假的值，PloarBear.html 也会在原窗口被打开。

```
<H2><A HREF="PolarBear.html"
onClick="openPolarBear();return false;">
Click here to see the polar bear.</A></H2>
```

8．输入<BODY>和<HTML>的结束标签。

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

9．将文件保存在学习磁盘的 Tutorial.05 文件夹中，文件名为 PolarBearMain.html。

下面创建 PolarBear.html 文档：

1．打开文本编辑器或 HTML 编辑器创建一个新文档。

2．在文档中输入开头的若干标签。

```
<HTML>
<HEAD>
<TITLE>Polar Bear</TITLE>
</HEAD>
```

3．输入开始 HTML 文档正文的<BODY>标签。

4．输入下面的标题标签。

```
<H1>This is a Polar Bear</H1>
```

5．添加一个<IMG>标签显示一幅名为 PolarBear.jpg 的图片。该图片的拷贝在学习盘的 Tutorial.05 文件夹中。

```
<IMG SRC="PolarBear.jpg">
```

6．然后添加下面的<FORM>和<INPUT>标签，它们用来创建一个按钮。当单击该按钮，就调用 self.close()方法关闭 PolarBear.html 文档。注意，还可以用 window.close()来执行相同的任务。

```
<FORM>
```

```
<INPUT TYPE=button NAME="quit"
VALUE=" Click here to close this window "
onClick="self.close();">
</FORM>
```

7．输入<BODY>和<HTML>的结束标签：

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

8．文件保存在学习磁盘的 Tutorial.05 文件夹中，文件名为 PolarBear.html。

9．在 Web 浏览器中打开 PolarBearMain.html 文件，然后单击链接“ Click here to see the polar bear ”。看到的显示结果应该和图 5-8 的显示一样。

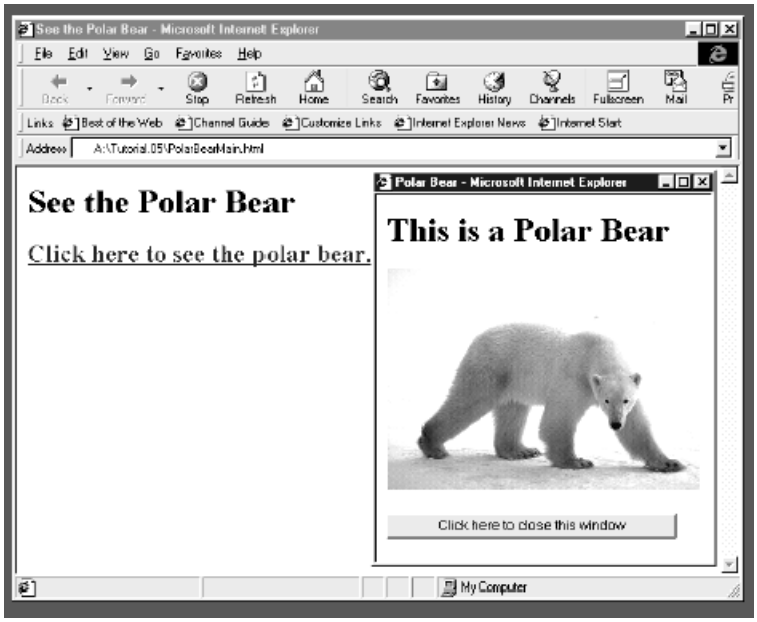


图 5-8　PolarBearMain.html 和 PolarBear.html 窗口

10．在新窗口中单击按钮“ Click here to close this window ”，关闭窗口，然后将原窗口也关闭掉。

提示：一些显示器的设置可能导致在 PolarBear.html 中看不到 Click here to close this window 按钮，如果看不到该按钮的话，尝试着调整 windows.open()方法的高度和宽度参数。

5.1.4　使用超时设定和间隔设定

在设计网页时，可能会需要一些不要用户干涉，自动重复执行的 JavaScript 代码。同

样地，可能需要创建动画或一些其他类型的任务，这些任务需要自动执行，不需要干涉。可以利用窗口对象的超时设定和间隔设定来编写代码，可以完成这些自动执行的任务。在 JavaScript 代码中，窗口对象的 setTimeout()方法完成这样的功能。它可以在过了指定的时间后，再执行代码。用 setTimeout()方法执行的代码仅执行一次。它的语法是：var variable =setTimeout("code ", milliseconds);。此处的变量声明将对 setTimeout()方法的引用赋给一个变量。其中的 code 参数必须用单引号或双引号引起来，可以是一行 JavaScript 语句，或者是一段语句体，还可以是一个函数调用。Web 浏览器需要等待指定的时间后，才执行 code 参数表示的代码，而这段时间是用参数 milliseconds 来表示的。milliseconds 是一秒的千分之一，五秒钟等于 5000milliseconds。而窗口对象的另一个方法 clearTimeout()用来在 setTimeout()方法的代码执行之前取消其执行。clearTimeout()只有一个参数，该参数是一个标示了 setTimeout()方法调用的变量。

图 5-9 显示了一个有 setTimeout()方法调用和 clearTimeout()方法调用的程序。setTimeout()方法的调用包含在 HTML 文档的<HEAD>部分，设定为等待 10000 毫秒后执行代码。如果用户单击了“ OK ”按钮，则函数 buttonPressed()将调用 clearTimeout()方法取消执行。

```
<HTML>
<HEAD>
<TITLE>Timeouts</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
var buttonNotPressed = setTimeout(
"alert('You must press the OK button to continue!')",
  10000);
function buttonPressed() {
clearTimeout(buttonNotPressed);
alert("The setTimeout() method was canceled!");
}
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
<BODY>
<FORM>
<INPUT TYPE=button NAME="OK" VALUE=" OK "
onClick="buttonPressed();">
</FORM>
</BODY>
</HTML>
```

图 5-9 一个使用了 setTimeout()方法和 clearTimeout()方法的程序

窗口对象的另两个相对的方法是 setInterval()和 clearInterval()。setInterval()方法和 setTimeout()方法类似。二者不同的地方是，后者只执行代码一次，而前者会重复执行代码。clearInterval()方法和 clearTimeout()方法类似，它用来取消 setInterval()方法的调用。setInterval()和 clearInterval()通常用来启动和重复执行一段动画代码。setInterval()的语法和 setTimeout()的语法是相同的：var variable = setInterval ("code ",milliseconds);。同样，clearInterval()函数也只接受一个参数，那就是代表了 setInterval()调用的变量。将会在第 7 章学习用 setInterval()和 clearInterval()方法创建动画。

下面将尝试往 PolarBearMain.html 文件中添加 setTimeout()方法。如果用户在 10 秒之内没有单击 polar bear 的链接，则用该方法自动调用 confirmPolarBear()方法。

PolarBearMain.html 文件中添加 setTimeout()方法：

1．在文本或 HTML 编辑器中打开 PolarBearMain.html 文件。

2．在函数 openPolarBear()前面声明一个全局变量 var polarBearOpened;。一个 setTimeout()方法的引用会被赋值给该变量。就像在第 2 章里学习到的，全局变量不同于局部变量，它的作用域是 JavaScript 程序的所有部分。

3．在函数 openPolarBear()后面添加下面的 confirmPolarBear()函数。该函数会在等待时间过后被调用。语句 confirm()用来提醒用户打开 PolarBear.html 文件。

```
function confirmPolarBear() {
        var confirmation =
         confirm("Do you want to see the polar bear or not?");
        if (confirmation == true)
        window.open('PolarBear.html', 'PolarBear',
        'height=350,width=320');
}
```

4．在<BODY>标签的后括号处添加一个空格，然后在空格后添加下面的时间设定语句：onLoad="var polarBearOpened=setTimeout('confirmPolarBear()',10000);"。如果在 10 秒钟之内用户并未动作，则该语句调用 confirmPolarBear()函数。

5．最后，如果用户单击了 polar bear 链接，则必须取消超时设置，要做到这点，只需在 openPolarBear()函数中的 windows.open()语句前添加：clearTimeout(polarBearOpened);。

6．保存文件，然后在 Web 浏览器中打开文件。等待 10 秒钟，看看确定对话框是否会弹出来，提示打开 PolarBear.html 文件。当看到确定对话框时，请单击“ OK ”按钮保证代码工作正确。

7．关闭 Web 浏览器窗口。

5.1.5 总结

- ✧ 窗口对象代表了一个 Web 浏览器窗口或窗口中一个单独的帧。
- ✧ JavaScript 对象的体系称为 JavaScript 对象模型。理解 JavaScript 对象模型的体系是十分重要的，因为一个对象的后代同时也是这个对象的属性。
- ✧ 在 JavaScript 对象模型里，可以用对象的属性和方法操纵窗口、帧和在 Web 浏览器中显示的 HTML 元素。
- ✧ 在 JavaScript 对象模型中，一个重要的对象是文档对象，它代表了显示在窗口中的文档。
- ✧ 虽然在本书中涉及到的 JavaScript 对象模型中的对象都是首字母大写。但是请注意，在编写代码时，必须把对象名称中的首字母用小写，因为实际用的是对象的一个属性。
- ✧ 当在代码中需要列出一个对象的祖先时，没有必要将窗口对象列出来，因为 Web 浏览器自动地认为操作是针对当前窗口进行的，在 JavaScript 对象模型里，窗口对象是处于体系的最高层的对象。当然，在某些情况下，必须包括窗口对象，例如在需要明确的区分窗口对象和文档对象的时候。
- ✧ 可以使用窗口对象的 open()方法打开一个新的 Web 浏览器窗口。
- ✧ 可以利用窗口对象的名称属性在超文本链接中标示一个目标窗口。
- ✧ 在 open()方法中，URL 参数表示将要打开的 Web 地址或文件名称。name 参数则给新窗口对象的名称属性指定一个名称。options 参数通过选项字符串控制新窗口的显示方式。
- ✧ 如果忽略了 open()方法的 options 参数，则 Web 浏览器将以常用的选项打开新窗口。如果在 open()方法中包括了 options 参数字符串，则请注意要将所有需要的选项都加以设置。否则，新窗口会只按照指定的选项进行显示，未指定的选项将会被忽略。
- ✧ 一个窗口对象的名称属性只能用来标示一个超文本链接中的窗口或表单，不能在 JavaScript 代码中使用。如果需要在代码中控制 Web 浏览器中打开的新窗口，那么需要在调用 open()方法时，将新窗口对象指定给一个变量。
- ✧ 通常我们在 open()和 close()方法前标明窗口对象，以区分是窗口对象还是文档对象的方法。
- ✧ 在 JavaScript 中，窗口对象的 setTimeout()方法用来等待指定的时间后再执行代码。
- ✧ 窗口对象的 clearTimeout()方法用来在 setTimeout()的代码执行前取消执行。
- ✧ 窗口对象的 setInterval()方法用来在调用后重复执行相同的代码。
- ✧ 窗口对象的 clearInterval()方法用来取消 setInterval()方法的调用。

5.1.6　问题

1．窗口对象代表一个：

a. <SCRIPT>…</SCRIPT>标签对

b. HTML 文档

c. Web 浏览器的窗口

d. <HEAD>…</HEAD>标签对

2. 下面的哪个名称不是指 JavaScript 对象模型？

a. 浏览器对象

b. 客户端对象

c. 导航器对象

d. IE 对象

3. 文档对象代表：

a. 表单中的一个对象

b. 在一个窗口中显示的 HTML 文档

c. 在一个 HTML 文档中所有的 JavaScript 函数和方法

d. Web 浏览器窗口

4. 对 JavaScript 代码中的文档对象，下面的哪条语句的语法是正确的？

a. document.writeln("text string");

b. Document.writeln("text string");

c. document.writeln("text string");

d. doc.writeln("text string");

5. 如果需要取得名为 inputField 的文本输入域的值，该文本输入域在名为 application 的表单中，下面哪个语句的语法正确？

a. inputField.value

b. application.inputField.value

c. document.application.inputField.value

d. application.value.inputField

6. self 属性是指：

a. 一个拥有焦点的控件

b. 当前执行的函数

c. 当前的文档对象

d. 当前的窗口对象

7. 在 window.open()方法的调用中，如果第一个参数是空串，将会发生什么情况？

a. 打开一个空白窗口

b. 收到一个出错消息

c. 打开一个当前窗口的完全拷贝

d. 什么也不做，JavaScript 忽略掉该语句

8. 在调用 window.open()方法后，下面的哪条参数串会在新窗口中包含水平和垂直滚动条？

a. verticalscroll=1,horizontalscroll=1

b. showScrollbars=on

c. scrollbars=true

d. scrollbars=yes

9. 如果忽略掉 window.open()方法的 options 参数，则_____。

a. 会创建一个带有菜单条和工具条的新的 Web 浏览器窗口

b. 会创建一个没有任何交互元素的空白窗口

c. 会创建一个包含常用选项的浏览器窗口

d. 会收到一个出错信息

10. 在 setTimeout()方法里，等待的时间长度单位按照_____计算。

a. 毫秒

b. 秒

c. 分钟

d. 10 秒间隔

11. setTimeout()方法可以用_____方法取消。

a. cancelTimeout()

b. clearTimeout()

c. endTimeout()

d. 没法取消调用了的 setTimeout()方法

12．在 setInterval()方法调用后，代码会被自动重复执行多少次？

a. 一次

b. 二次

c. 重复执行

d. 0 次

13．setInterval()方法可以用_____方法取消。

a. cancelInterval()

b. clearInterval()

c. endInterval()

d. 没法取消调用了的 setInterval()

5.1.7 练习

1. 创建一个 HTML 文档，该文档包含一个您喜欢的菜谱的超链列表。单击每个链接将

在另一个单独的窗口中打开一个文档，该文档显示选择的菜谱。在每个窗口中都包含一个“关闭”按钮。将主 HTML 文档命名为 Recipes.html，根据每个菜谱的名称给它们的文档命名。例如，如果您有一份苹果派的菜谱，就将和它相关的文档命名为 ApplePie.html。将所有文件保存在学习盘的 Tutorial.05 文件夹中。

2. 创建一个在线应用，在该应用中提示用户输入它们的一些个人信息，例如姓名、地址、城市、邮编、出生日期等。用一个提示对话框收集每条信息，并将信息存在单独的变量名里。用确认对话框提示用户是否需要显示他们输入的信息。如果用户选择“yes”，则在一个单独的 alert 对话框中显示所有变量。用换行符分隔每个变量。在显示变量之前，请用一个决策结构检查它是否为空值。如果为空值，则将其忽略。将文件保存在学习盘的 Tutorial.05 文件夹中，文件名称为 VitalInfo.html。

3. 创建一个 HTML 文档，该文档将状态条上的一条默认状态信息重复的显示和隐藏，使效果和一个闪烁的霓虹灯相似。可以利用类似于 if…then 语句的决策结构创建文档。将文件保存在学习盘的 Tutorial.05 文件夹中，文件名称为 FlashGreeting.html。

4．创建一个可以让用户玩一个猜谜游戏的 HTML 文档。出一个数字，然后赋给一个变量。用<INPUT>标签在表单上创建一个文本输入域，由用户在该处输入猜测的数字。用<INPUT>标签创建另一个名为“Guess”的按钮。设定一个时间限制，如果用户在 10 秒后没有单击“Guess”按钮，则提示用户是否还继续游戏。如果用户选择“取消”，则关闭 Web 浏览器窗口。如果用户选择“继续”，则重新启动游戏和时间限制。将文件保存在学习盘的 Tutorial.05 文件夹中，文件名称为 GuessNumber.html。

5.2　使用帧和其他对象

本节目标

在本节里将学习：

- ✧ 如何创建帧
- ✧ 如何使用 TARGET 属性
- ✧ 如何安排帧
- ✧ 关于 NOFRAMES 标签的知识
- ✧ 关于定位对象的知识
- ✧ 关于历史对象的知识
- ✧ 关于导航器对象的知识
- ✧ 如何引用帧和窗口

5.2.1　创建帧

到目前为止，创建的 HTML 文档都是由一个窗口对象组成的，而且一次只能保留一个

URL。利用帧结构，可以将一个 Web 浏览器的窗口划分成多个窗口，每个窗口可以打开一个不同的 URL。帧是一个独立的、可滚动的 Web 浏览器窗口区域，每个帧都可以保留自己的 URL。JavaScript 将一个 HTML 文档里的每帧都作为独立的窗口处理。每个独立的帧都有自己的窗口对象，要和文档中的其他帧区别开来。此外，每帧都是定义它们的顶层 HTML 文档的一部分。每帧的窗口对象都是从顶层 HTML 文档包含的窗口对象继承来的。虽然帧是 HTML 的元素，实际上并不是 JavaScript 创建的，JavaScript 还是可以用程序存取或控制单独的帧。

可以用<FRAMESET>…</FRAMESET>标签对将一个 HTML 文档分成若干帧。<FRAME>和其他的<FRAMESET>…</FRAMESET>标签对是惟一可以放置在<FRAMESET>…</FRAMESET>标签对里面的项目。Web 浏览器忽略任何其他的文本和标签。<FRAMESET>…</FRAMESET>标签对替代了在没有帧的 HTML 文档中的<BODY>…</BODY>标签。要注意的是，在一个包含了<FRAMESET> … </FRAMESET>标签对的 HTML 文档中不能再包括<BODY>标签。如果包括了<BODY>标签，则<FRAMESET>标签会被忽略掉。

一个 HTML 文档中的帧可以水平划分、垂直划分，也可以在水平和垂直方向上都划分。<FRAMESET>标签的两个属性：ROWS 和 COLS，决定了是按照行还是列来划分帧。ROWS 决定在水平方向上创建多少帧，COLS 决定在垂直方向上创建多少帧。要确定帧的尺寸，需要赋给 ROWS 或 COLS 属性一个字符串，该字符串包含了行和列上的每帧尺寸应该占屏幕的百分比，或者像素值，并且用逗号将其隔开。例如，<FRAMESET ROWS="50%,50%" COLS="50%, 50%">创建两行两列、4 个帧，每帧的高度为屏幕高度的 50%，每帧的宽度为屏幕宽度的 50%。图 5-10 显示了这个例子的结果。

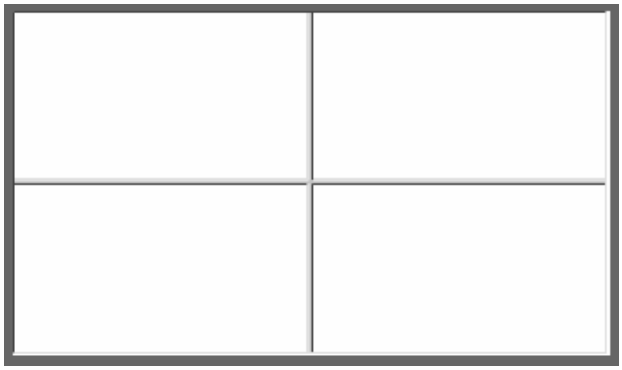


图 5-10 用语句<FRAMESET ROWS="50%,50%" COLS="50%,50%">创建的帧

提示：必须定义超过一列或一行的帧，否则 Web 浏览器将会完全忽略创建的帧。

可以仅在行或列上创建帧。例如，图 5-11 显示了用<FRAMESET ROWS="50%,50%">创建的帧，图 5-12 显示了用<FRAMESET COLS="50%,50%">创建的帧。

图 5-11　用语句<FRAMESET ROWS="50%,50%">创建的帧

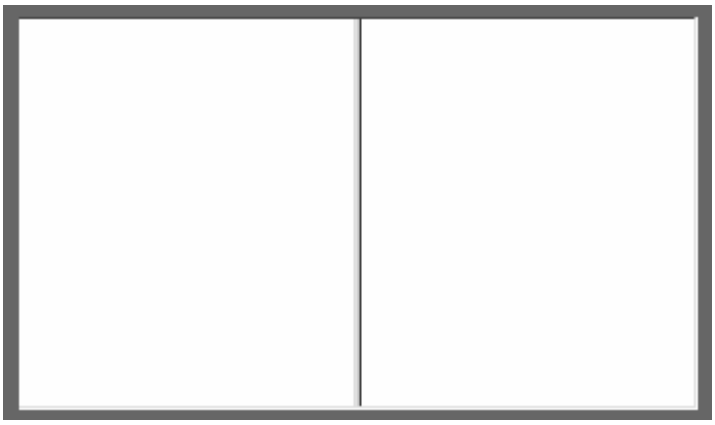


图 5-12　用语句<FRAMESET COLS="50%,50%">创建的帧

当用百分比标示帧的尺寸时，这里的百分比是相对于窗口的大小来说的。也就是说，会随着窗口而调整。相反，像素表示精确或实际的尺寸。它们不会随着窗口尺寸调整。用户可以为它们的 Web 浏览器窗口设置不同的默认尺寸。用相对的百分比可以充分考虑这些变化，而用像素就做不到这个要求。例如，如果创建两个帧，每个 100 像素宽度，则它们不会随着用户 Web 浏览器窗口的实际大小进行调整。如果浏览器窗口太小，帧可能被遮盖掉；如果浏览器窗口太大，帧可能显得格外小。如果用百分比，则这些问题就不会成为问题，因为帧的大小会随着 Web 浏览器窗口大小重新计算调整。

用星号*来表示帧的尺寸是非常有用的，这样在文档里，就不必非要指定一个精确的像素值或百分比。星号将所有保留的屏幕空间都分配给一个单独的帧。如果有一个以上的帧使用了星号，则将所有保留的屏幕空间平均分配。例如，<FRAMESET COLS="100,*"> 创建分成两列的两个帧，其中一列用像素表示，而星号就代表另一列。左边列的帧将一直保持 100 像素的宽度，而右边列的帧会根据当前屏幕可用的空间进行调整。

还可以把像素、百分比和星号结合起来用。举例来说，下面的标签<FRAMESET

ROWS="100, 50%,*"创建三个水平的帧：第一个帧 100 像素高，第二个帧占据可用窗口空间的 50%，星号将余下的空间安排给了第三个帧。

标签<FRAMESET>用来在 HTML 文档中创建起始帧。标签<FRAME>则用来创建指定选项的独立帧，包括帧的 URL。该标签的 SRC 属性指明了在该单独帧中将要打开的 URL。FRAME 标签被放置在<FRAMESET>…</FRAMESET>标签对之内。可以用名称属性给帧指定一个名称，该名称可以在超级链接中用做链接目标。

帧的 URL 按照在<FRAMESET>…</FRAMESET>标签对内<FRAME>排列的顺序打开，该顺序是从左到右，从上到下。例如，下面的代码创建 4 个帧，分两行和两列排列。图 5-13 显示了用<FRAME>标签的 URL 是按照怎样的顺序打开的。在每个帧中显示的文本包含在 SRC 属性标示的文件中。

```
<FRAMESET ROWS="50%, 50%" COLS="50%, 50%">
<FRAME SRC="FirstURL.html">
<FRAME SRC="SecondURL.html">
<FRAME SRC="ThirdURL.html">
<FRAME SRC="FourthURL.html">
</FRAMESET>
```

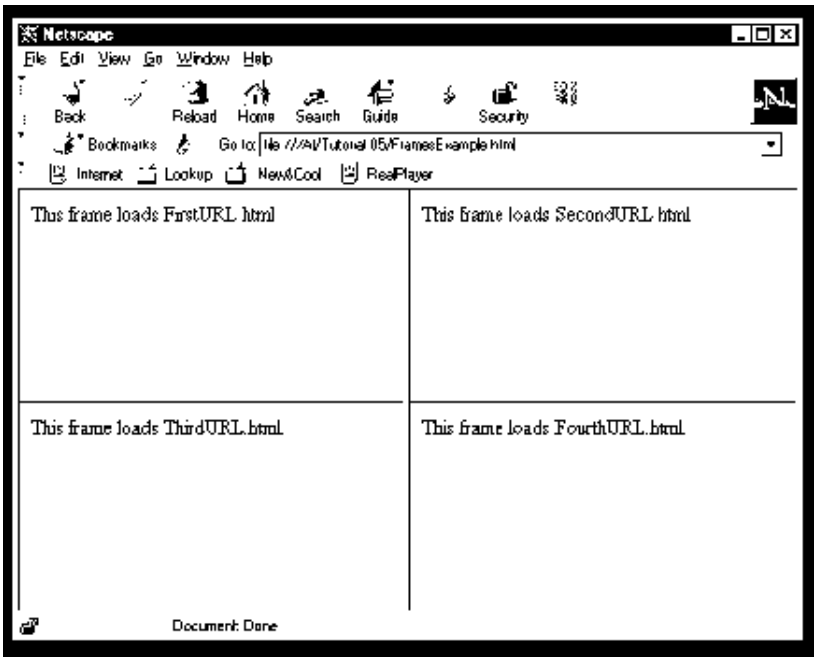


图 5-13 URL 的载入顺序

接下来，将开始创建本章前的预览中了解的虚拟动物园程序。首先会创建程序中包含<FRAMESET>和<FRAME>标签的主 HTML 文档。该程序创建一个狭窄的列，该帧包含动物名称的列表。右边较大的帧占据了余下的屏幕，用来显示动物图像。

创建虚拟动物园程序的主 HTML 文档：

1．启动文本或 HTML 编辑器建立一个新文档。

2．输入文档开始的标签。

```
<HTML>
<HEAD>
<TITLE>Virtual Zoo</TITLE>
</HEAD>
```

3．添加<FRAMESET COLS="20%, *">启动帧设置。代码中的 20%创建左边列中狭窄的帧，星号则创建右边列中的帧。

4．添加下面两个<FRAME>标签。第一个标签打开一个名为 list.html 的文件，该文件包含动物名称的列表。第二个帧打开一个名为 welcome.html 的 HTML 文件，该文件包含了一个显示信息。包含 list.html 窗口命名为 list，包含有 welcome.html 的窗口命名为 display。

```
<FRAME SRC="list.html" NAME="list">
<FRAME SRC="welcome.html" NAME="display">
```

5．输入下面的代码作为结束标签。

```
</FRAMESET>
</HTML>
```

6．将文件存到 Turtorial.05 文件夹，文件名称为 VirtualZoo.html。在打开 VirtualZoo.html 文件之前，必须先创建 welcome.html 和 list.html 文件。

下面创建 welcome.html 文件：

1．在文本或 HTML 编辑器中创建一个新的文档。

2．输入文档开始的标签。

```
<HTML>
<BODY>
```

3. 添加下面的代码，引导用户单击列表中的一个动物名称，并按回车：

```
Click an animal in the list to display its picture.
```

4．输入结束标签。

```
</BODY>
```

```
</HTML>
```

5．保存文件到 Tutorial.05 文件夹中，文件名称为 welcome.html。

5.2.2 使用 TARGET 属性

帧的一个最常见的作用是，在左边的帧中显示内容的列表，而在右边的帧中显示左边列表的内容。这种类型的设计避免了在需要浏览其他 URL 内容时，必须打开另一个 Web 浏览器窗口的麻烦。就像在第 1 节里，在文件 PolarBear.html 中做的一样。图 5-14 显示了一个 HTML 文档，该文档被分成两个帧。每个帧显示一个链接在不同 URL 上的 HTML 文档。左边的帧显示了一个有乐器列表的 HTML 文档，右边的帧显示了一个提示用户选择一件乐器的 HTML 文档。一个新的包含有乐器图像和描述的文档将会在右边的帧打开，如图 5-15 所示。

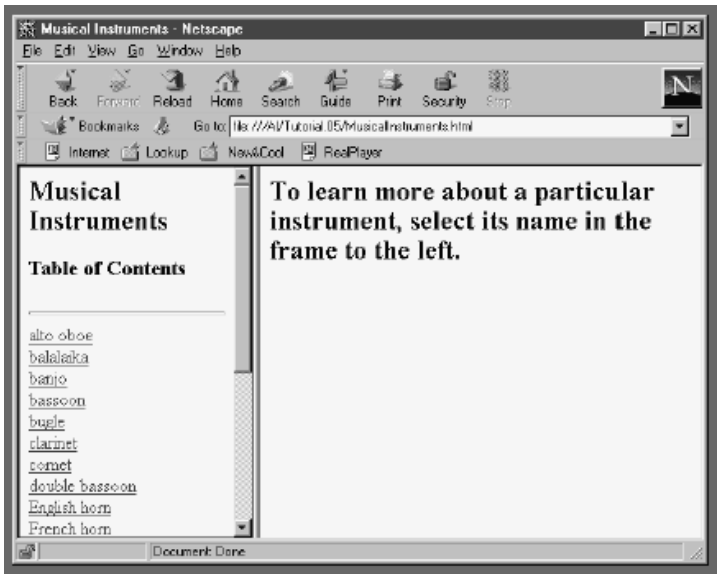


图 5-14　关于乐器的文档

图 5-16 显示了如何用<FRAMESET>和<FRAME>标签创建关于乐器的文档。

在乐器程序里，创建了两个列帧。第一列 200 个像素宽度，第二列占据了余下的窗口空间。每个帧内的 HTML 文档由两个<FRAME>标签打开。左边的帧包含了每件乐器名称的超链。可以利用<A>标签的 TARGET 属性在右边的帧中打开每个超链链接的 HTML 文档。TARGET 属性指定在哪个帧或 Web 浏览器窗口中打开 URL。例如，在乐器程序中，右边帧的名称为 display。当单击左帧的乐器-大号进行链接时，tuba.html 文件将在右帧中打开。实现该功能的语法是：

```
<A HREF="tuba.html" TARGET="display">;
```

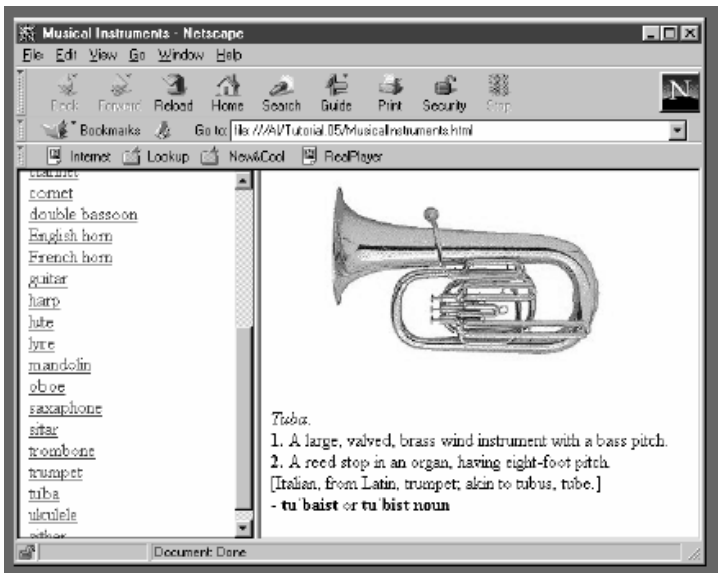


图 5-15　用户选择一件乐器后的文档

```
<HTML>
<HEAD>
<TITLE>Musical Instruments</TITLE>
</HEAD>
<FRAMESET COLS="200,*">
<FRAME SRC="InstrumentsList.html" NAME="list">
<FRAME SRC="Welcome.html" NAME="display">
</FRAMESET>
</HTML>
```

图 5-16　关于乐器文档的程序

当需要为一长列的超链指定目标窗口或帧时，推荐用<BASE>标签代替为每个链接重复键入 TARGET 属性。<BASE>标签用来在一个 HTML 文档中为所有的链接标示一个默认的目标，通常该目标是窗口或帧的名称。下面是一段重复地使用<TARGET>属性的代码：

```
<A HREF="altooboe.html" TARGET="display">alto oboe</A><BR>
<A HREF="balalaika.html" TARGET="display">balalaika</A><BR>
<A HREF="banjo.html" TARGET="display">banjo</A><BR>
<A HREF="bassoon.html" TARGET="display">bassoon</A><BR>
<A HREF="bugle.html" TARGET="display">bugle</A><BR>
其他乐器的语句
```

可以用<BASE>标签将该段程序改写得更有效率。

```
<BASE TARGET="display">
<A HREF="altooboe.html">alto oboe</A><BR>
<A HREF="balalaika.html">balalaika</A><BR>
<A HREF="banjo.html">banjo</A><BR>
<A HREF="bassoon.html">bassoon</A><BR>
<A HREF="bugle.html">bugle</A><BR>
其他乐器的语句
```

下面将学习创建 list.html。在虚拟动物园程序中，该文件包含动物名称的列表。

创建 list.html：

1. 启动文本或 HTML 编辑器创建一个新文档。
2. 输入文档开始的标签。

```
<HTML>
<BODY>
```

3. 由于每个动物的图像都将显示在右边的帧，所以添加<BASE TARGET="display">，将右边的帧设定为默认目标。

4．回车，然后为每个动物添加下面的链接。需要注意的是，此处不是在目标窗口打开 HTML 文档，而是打开 JPG 的图片文件（这些 JPG 图片文件保存在 Tutorial.05 文件夹中）。

```
<A HREF="Elephant.jpg">Elephant</A><BR>
<A HREF="Gazelle.jpg">Gazelle</A><BR>
<A HREF="Giraffe.jpg">Giraffe</A><BR>
<A HREF="Lion.jpg">Lion</A><BR>
<A HREF="PolarBear.jpg">Polar bear</A><BR>
<A HREF="Rhino.jpg">Rhino</A><BR>
<A HREF="Tiger.jpg">Tiger</A><BR>
<A HREF="Zebra.jpg">Zebra</A><BR>
```

5．输入结束标签。

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

6．将文件保存在 Tutorial.05 文件夹中，取名为 list.html。

7．目前为止，已经创建了 list.html 和 welcome.html 文件，可以在 Web 浏览器中打开 VirtualZoo.html 文件。单击每个动物的名称，然后看程序是否能够正确地工作。图 5-17 显示了程序在浏览器中显示长颈鹿图片的结果。

8．关闭 Web 浏览器窗口。

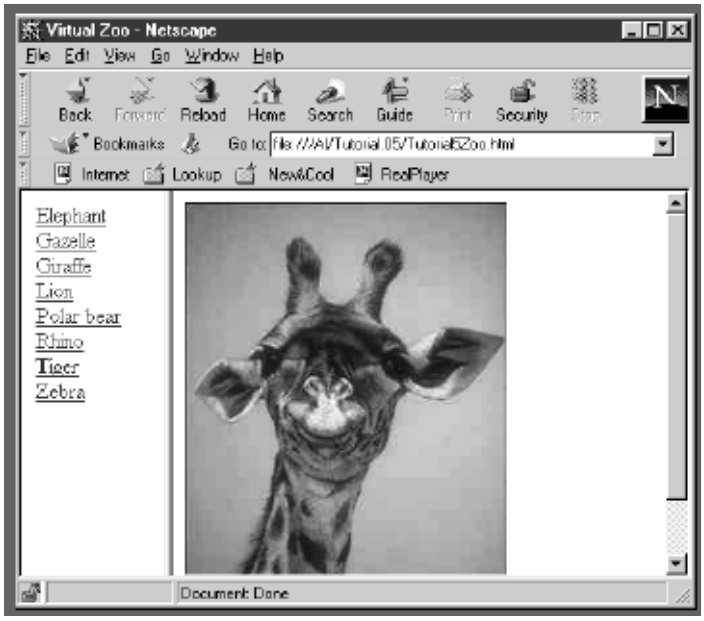


图 5-17　浏览器中的 VirtualZoo.html

5.2.3　帧的嵌套

JavaScript 将 HTML 文档中的每个帧看作一个单独的窗口。因此，窗口中的每个帧都可以包含自己的帧集合。可以通过将一个<FRAMESET>…</FRAMESET>标签对放在另一个<FRAMESET>…</FRAMESET>标签对内来实现这种嵌套。包含在其他帧里的帧，我们称其为嵌套的帧。

当 Web 浏览器创建帧时，帧的 URL 地址是按照<FRAME>标签的顺序载入的。下面的程序创建了一个包含两行和两列的父帧集合。在第二个帧里又创建了一个同样包含两行和两列的嵌套的帧集合。在每帧里显示的文本包含在由 SRC 属性指定的文件中。图 5-18 显示了程序的结果。

```
// The following line creates the main frame set
<FRAMESET ROWS="50%, 50%" COLS="50%, 50%">
    // The following line assigns frame1.html as the
    // URL of the first frame in the main frame set
    <FRAME SRC="frame1.html">
    // The following line creates a nested frame set
    // inside the second frame in the main frame set
    <FRAMESET ROWS="50%, 50%" COLS="50%, 50%">
```

```
        // The following lines assign URLs
       // to the nested frames
            <FRAME SRC="frame1.html">
            <FRAME SRC="frame2.html">
            <FRAME SRC="frame3.html">
            <FRAME SRC="frame4.html">
    </FRAMESET>
    // The following lines assign URLs to the
    // third and fourth frames in the main frame set
    <FRAME SRC="frame3.html">
    <FRAME SRC="frame4.html">
</FRAMESET>
```

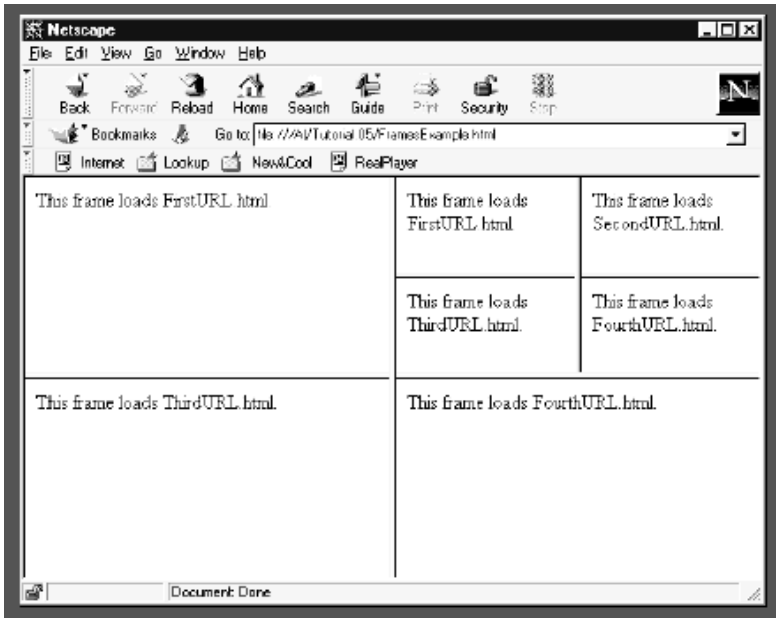


图 5-18　嵌套的帧

在上面的例子中，第一个<FRAMESET>标签创建窗口中的四个父帧，第一个<FRAME>标签将 frame1.html 文件指定给父帧集合中的第一个帧，第二个<FRAMESET>标签被嵌套在父集合的第二个帧里，每个嵌套的帧都指定了一个 URL 地址。在图 5-18 中显示的嵌套的帧要比我们平常见的页面复杂的多。

下面的代码显示了一个更典型的例子，该示例使用乐器程序。右边的父帧包含了嵌套的、分成两行的帧集合。第一个子帧显示乐器的图像，第二个子帧显示乐器的说明。图 5-19 显示了该程序的结果。

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

```
<TITLE>Musical Instruments</TITLE>
</HEAD>
<FRAMESET COLS="200,*">
    <FRAME SRC="left.html" NAME="list">
    <FRAMESET ROWS="75%,*">
        <FRAME SRC="instruments.jpg" NAME="picture">
        <FRAME SRC="welcome.html" NAME="description">
    </FRAMESET>
</FRAMESET>
</HTML>
```

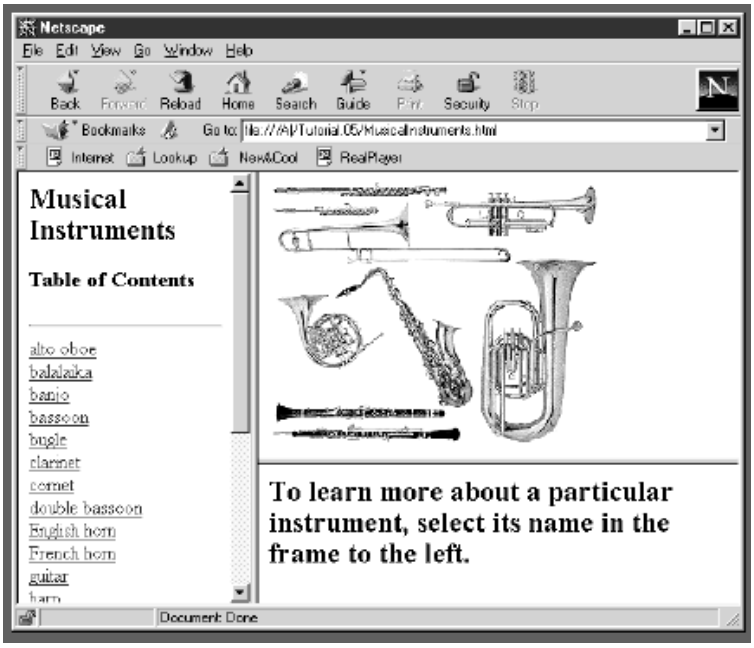


图 5-19　利用嵌套帧的乐器程序

下面将尝试修改虚拟动物园程序，使该程序包含嵌套的帧。父帧集合分成两行：第一行显示虚拟动物园的标题；第二行将包含一个嵌套的帧集合，由动物的名称列表和动物图像两个帧构成。

给虚拟动物园程序添加嵌套的帧：

1. 在文本或 HTML 编辑器中打开 VirtualZoo.html 文件。

2. 在原有的<FRAMESET>标签前添加<FRAMESET ROWS="20%,*">。原有的帧集合将被嵌套在新的帧集合中。

3. 在新的帧集合的开始标签后面，添加<FRAME SRC="title.html" NAME="title">，指定在第一个帧中打开名为 title.html 的文件。

4. 在</HTML>标签前添加</FRAMESET>来关闭新的帧集合。

5. 保存并关闭文件。

然后创建 title.html 文件：

1. 在文本或 HTML 编辑器中创建一个新文档。

2. 输入文档开始的标签：

```
<HTML>
<BODY>
```

3．添加<H1>Virtual Zoo</H1>，以创建一个标题 1 格式的标题行。

4．输入结束标签。

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

5. 将文件保存到文件夹 Tutorial.05，名称为 title.html，在 Web 浏览器中重新打开文件 VirtualZoo.html。图 5-20 显示了选择瞪羚后程序的结果。

6．关闭 Web 浏览器窗口。

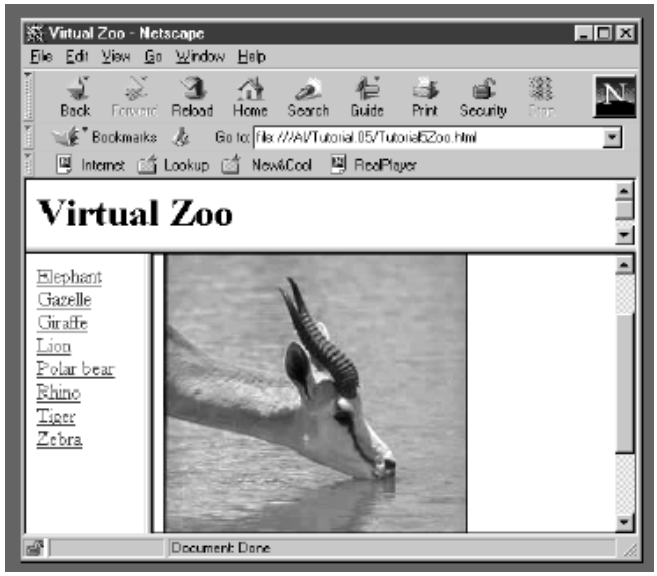


图 5-20　使用嵌套的帧的 VirtualZoo.html

5.2.4　帧的格式

<FRAME>标签包含了几个可以改变帧的显示格式和行为的属性。表 5-4 列出了这几个属性。

表 5-4　<FRAME>标签的属性

属　　性	描　　述
SRC	指定在一个帧中要打开的 URL 地址
NAME	给帧指定一个名称
NORESIZE	禁止用户调整一个单独帧的尺寸
SCROLLING	设置一个帧是否包含滚动条
MARGINHEIGHT	以像素为单位，指定帧的顶部和底部空白
MARGINWIDTH	以像素为单位，指定帧的左边和右边空白

已经用过 SRC 属性为帧指定一个 URL 地址。也学习了用 NAME 属性指定一个帧为超文本链接的目标。NORESIZE 属性使用户不能再重新调整一个单独帧的大小。如果需要添加一个标题，该标题要求总是显示在帧或页面上；或者需要在页面的底部建立一些链接的列表，以方便用户浏览您的站点。通常在这些情况下，才使用 NORESIZE 属性。更常见的情况是，用户常常由于自己的需求要调整帧的尺寸。如果不允许用户调整帧的尺寸，就需要在<FRAME>标签里添加 NORESIZE 属性。

默认情况下，浏览器自动地为帧添加滚动条，如果浏览的内容超过了可见的屏幕区域的话。可以通过 SCROLLING 属性将滚动条屏蔽掉。SCROLLING 属性可以取三种值：yes、no 和 auto。取值为 yes 将一直打开滚动条（即使内容全在可见区域内的话也会带着滚动条）；取值为 no 将取消滚动条（即使内容和可见区域并不十分合适）；取值为 auto，则根据帧中内容和可见区域的大小自动调整滚动条。选择 auto 的效果等同于在<FRAME>标签中不包括 SCROLLING 属性。

下面的代码显示了一个带有 NORESIZE 和 SCROLLING 属性的例子。图 5-21 显示了该程序的输出结果。该程序包含了三个帧。最顶层的帧给页面提供了一个标题，最底层的帧包括了导航按钮和链接。中间的帧包含了页面的主要内容。因为我们不想让用户调整顶层和底层的帧的尺寸，所以在这两个帧的<FRAME>标签里包括了 NORESIZE 属性。同时，也没有为这两个帧设置滚动条属性，因为用户没有必要在顶层和底层进行滚动。

```
<FRAMESET ROWS="20%, *, 20%">
    <FRAME SRC="header.html" NORESIZE SCROLLING=no>
    <FRAME SRC="body.html">
    <FRAME SRC="navigationbar.html" NORESIZE SCROLLING=no>
</FRAMESET>
```

提示：在图 5-21 程序的输出中，顶层帧和底层帧的 NORESIZE 属性就限定了中间的帧也没办法调整尺寸。

MARGINHEIGHT 和 MARGINWIDTH 属性决定了帧以像素为单位的空白的大小。图 5-22 显示了图 5-21 的程序在调整中间的帧的 MARGINHEIGHT 为 50，MARGINWIDTH 为 50 后的输出。调整的标签为 <FRAME SRC="body.html" MARGINHEIGHT=50

MARGINWIDTH=50>。

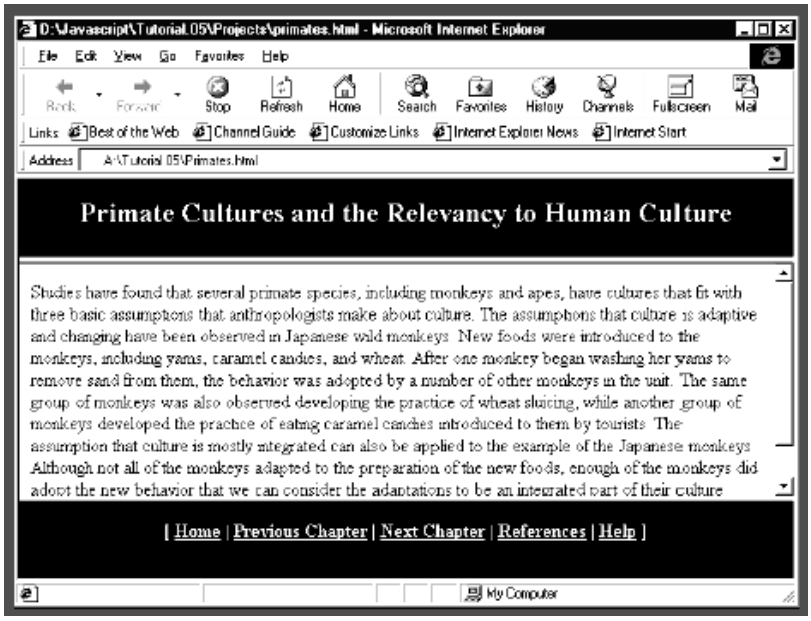


图 5-21　包含 NORESIZE 和 SCROLLING 属性的程序输出

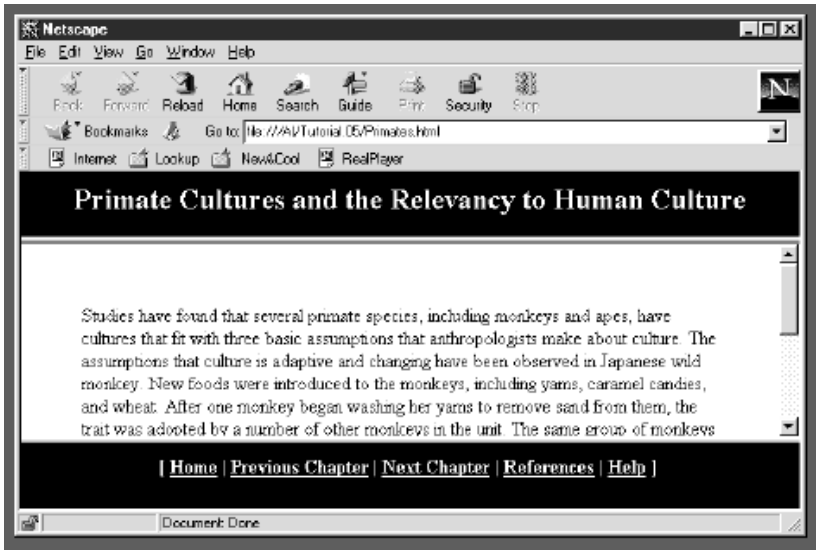


图 5-22　调整空白后的 5-21 程序的输出

下面添加 NORESIZE 和 SCROLLING 属性到虚拟动物园程序的标题帧中：

1. 在文本或 HTML 编辑器中打开 VirtualZoo.html 文件。

2. 在<FRAME　SRC="title.html"　NAME="title">标签的右括号前添加 NORESIZE 和 SCROLLING =no。

3. 保存并关闭文件，然后在浏览器中打开 VirtualZoo.html。在标题帧中，已经没有了滚动条，而且也不能再调整它的尺寸了。

5.2.5　NOFRAMES 标签

就像一些老的 Web 浏览器不支持 JavaScript 一样，它们也不支持帧结构。在第 1 章里，阐述了如何用<NOSCRIPT>…</NOSCRIPT>标签对给用户显示一条 Web 浏览器不支持 JavaScript 的信息。标签<NOFRAMES>…</NOFRAMES>完成类似的功能。它在不支持帧结构的浏览器中显示给用户一条替代信息。<NOFRAMES>…</NOFRAMES>标签对通常跟在<FRAMESET>…</FRAMESET>标签对的后面。下面是一段例子程序：

```
<FRAMESET ROWS="20%, *, 20%">
    <FRAME SRC="header.html" NORESIZE SCROLLING=no>
    <FRAME SRC="body.html">
    <FRAME SRC="navigationbar.html"
        NORESIZE SCROLLING=no>
</FRAMESET>
<NOFRAMES>
You cannot view this Web page because your Web browser
does not support frames. To view a no frames version of this
Web page,click <A HREF="no_frames.html">here</A>
</NOFRAMES>
```

提示：支持帧结构的浏览器将忽略掉<NOFRAMES>标签。

下面添加<NOFRAMES>标签到虚拟动物园程序中：

1. 在文本或 HTML 编辑器中打开 VirtualZoo.html。

2. 在最后一个</FRAMESET>标签后，加入<NOFRAMES>标签对，以便在浏览器不支持帧结构时提醒用户。

```
<NOFRAMES>
    You cannot view this Web page because your Web browser
    does not support frames.
</NOFRAMES>
```

3. 保存并关闭文件。现在，如果有用户使用不兼容帧结构的浏览器打开 VirtualZoo.html 文件，将会看到 NOFRAMES 中的消息。

5.2.6 定位对象

当需要允许用户从一个网页中打开另一个网页时，通常要用<A>标签创建超文本链接。还可以用 JavaScript 代码和定位对象来打开一个网页。定位对象允许用 JavaScript 语句改变到另一个 Web 页面。需要变化 Web 页面的原因可能是要重新定位用户在站点内的浏览，可能是要将用户导向一个不同的或更新后的 URL 地址。定位对象包含若干属性和方法，这些属性和方法都是用来操作浏览器当前窗口中的 URL 文档的。当使用这些属性和方法时，必须包含一个对定位对象的引用。举例来说，如果要利用 href 属性，必须写成 location.href=URL;。表 5-5 列出了定位对象的属性。

表 5-5 定位对象的属性

名 称	描 述
hash	一个 URL 的锚链
host	URL 的主机名和端口的组合
hostname	一个 URL 的主机名称
href	一个完全的 URL 地址
pathname	URL 的路径
port	URL 的端口
protocol	URL 的协议
search	一个 URL 的搜索或查询部分

定位对象的属性允许修改一个 URL 地址的单独的部分。当修改定位对象的任何属性时，就创建了一个新的 URL 地址，浏览器自动地打开这个地址。通常情况下，并不采用修改 URL 地址的单独部分，而是改动 href 属性，它代表了整个的 URL 地址。例如，语句 location.href="http://www.netscape.com";会将网景公司的主页打开。

定位对象包括两个方法：reload()和 replace()。reload()方法相当于 Navigator 里的重新载入按钮或 IE 里的重新刷新按钮。它重新打开浏览器中当前显示的页面。在使用 reload()方法时，可以不要任何参数，如 location.reload();，或者可以包含一个布尔型的参数。当该参数为真时，强迫浏览器从当前页面所在的服务器重新载入，即便网页没发生任何变化。举例来说，语句 location.reload(true);强制重新载入页面。如果参数为假，或者不包括任何参数，则浏览器只有在页面发生改变时才重新载入。

定位对象的 replace()方法用另一个不同的 URL 地址替换当前的 URL。该方法与通过改变 href 属性载入一个新文档有所不同。replace()方法实际上是用另一个文档重写了当前文档，并且用新的 URL 地址顶替了原地址在浏览器历史记录中的项目。相反，href 属性打开一个新文档，并且将其添加到历史记录中。将在下节学习历史记录。

5.2.7　历史对象

历史对象包含了在当前浏览器链接期间曾经打开文档的历史记录。不管打开了多少个窗口或帧，每个 Web 浏览器窗口或帧都有自己内部的历史对象。用 JavaScript 程序操作历史记录，可以转向任何在浏览器会话期间曾经打开的页面。如果创建了用户导航控件，例如表单按钮，则可以用历史对象进行程序性控制。历史对象包括 3 个方法，列在表 5-6 中。

表 5-6　历史对象的方法

方　法	描　述
back()	该方法等同于单击 Web 浏览器的后退按钮
forward()	该方法等同于单击 Web 浏览器的前进按钮
go()	打开在历史记录中的一个指定文档

提示：本书只描述在 Navigator 4 和 IE 4 中都兼容的历史对象的方法和属性。

当用历史对象的方法和属性时，必须包括对历史对象的一个引用。例如，要用 back()方法，应该写成 history.back()。

back()和 forward()方法允许程序在浏览器的历史记录中向前或向后移动。图 5-23 中的代码演示了该如何利用 back()和 forward()方法，控制在一个名为 samplePages 的窗口中显示的页面。该窗口中的页面已经被改变了若干次，创建了一个历史记录列表。源代码中包含了模拟 Web 浏览器中的前进和后退按钮的两个按钮。单击这两个按钮就可以改变在 samplePages 窗口中显示的页面。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
samplePages = window.open("FirstWebPage.html", "WebPages");
samplePages.location.href = "SecondWebPage.html";
samplePages.location.href = "ThirdWebPage.html";
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
<BODY>
<FORM>
<INPUT TYPE=button NAME="Back" VALUE=" Back "
      onClick="samplePages.history.back();">
<INPUT TYPE=button NAME="Forward" VALUE=" Forward "
      onClick="samplePages.history.forward();">
</FORM>
</BODY>
```

图 5-23　运用历史对象的 back()和 forward()方法的程序

历史对象中的 go()方法用来转向一个以前曾经浏览过的指定页面。该方法的参数是一个整型数，该整型数表示要往前或往后转过的页面数目。举例来说，语句 history.go(-2);将打开在历史记录中后退 2 个页面的那个 Web 页面。语句 history.go(3);将打开在历史记录中前进 3 个页面的那个 Web 页面。语句 history.go(-1);等同于方法 back()，语句 history. go(1)等同于方法 forward()。go()方法还可以用曾经打开过的 Web 页面的 URL 地址字符串作为参数。如果只使用 URL 的一个部分或子串作为参数，则 go()方法打开最接近的历史记录。例如，下面的代码将打开 Course Technology 主页，通过语句 history.go("Course");。

```
location.href = "http://www.course.com";
location.href = "http://www.netscape.com";
history.go("Course");
```

在上面的例子里，如果不用语句 history.go("Course");，通过其他方法同样可以转向 Course Technology 的站点，可以用语句 history.go(-1)或 history.back();。

历史对象只有一个属性，即长度属性，它包含了在当前浏览器会话期间已经被打开的文档的数目。使用 length 属性的语法是 history.length;。长度属性并不包含文档的 URL 本身，而是曾经打开过的文档的一个计数。下面的代码用一个提示对话框显示了曾经浏览过的页面数目。

```
location.href = "FirstWebPage.html";
location.href = "SecondWebPage.html";
location.href = "ThirdWebPage.html";
alert("You have visited " + history.length
+ " Web pages.");
```

5.2.8 领航员对象

领航员对象用来获得关于当前 Web 浏览器的信息。虽然 Navigator 对象从 Netscape Navigator 命名而来，但是在 IE 中同样得到了支持。领航员对象没有任何方法，只有在表 5-7 中列出的属性。

表 5-7 领航员对象的属性

属　　性	返　　回
appCodeName	Web 浏览器的代码名称
appName	Web 浏览器的名称
appVersion	Web 浏览器版本
language	Web 浏览器使用的语言，例如英语或法语
platform	当前使用的操作系统
userAgent	用户代理

IE 并不支持在表 5-7 中列出的 language 属性。相反，IE 用其他两种属性来替代：userLanguage 和 systemLanguage。

Netscape Navigator 和 IE 都有自己独有的、对方不兼容的方法和属性。例如，Netscape Navigator 包括一个 home()方法，该方法用来打开浏览器公司的主页。IE 就不支持 home()方法。如果尝试在 IE 中运行一个包含有 home()方法的 JavaScript 程序，会收到一个错误。正是由于这些不兼容性，程序员用领航员对象的属性来判断是哪种类型的浏览器在运行。语句 browserType = navigator.appName;返回代码正在运行的 Web 浏览器的名称，并赋给变量 browserType。然后可以在需要指定浏览器类型的情况下，利用该变量进行判断。

下面的语句将 Netscape Navigator 的领航员对象的 6 个属性打印出来。图 5-24 显示了输出结果。

```
with (navigator) {
    document.writeln("Browser code name: " + appCodeName);
    document.writeln("Web browser name: " + appName);
    document.writeln("Web browser version: " + appVersion);
    document.writeln("Language: " + language);
    document.writeln("Operating platform: " + platform);
    document.writeln("User agent: " + userAgent);
}
```

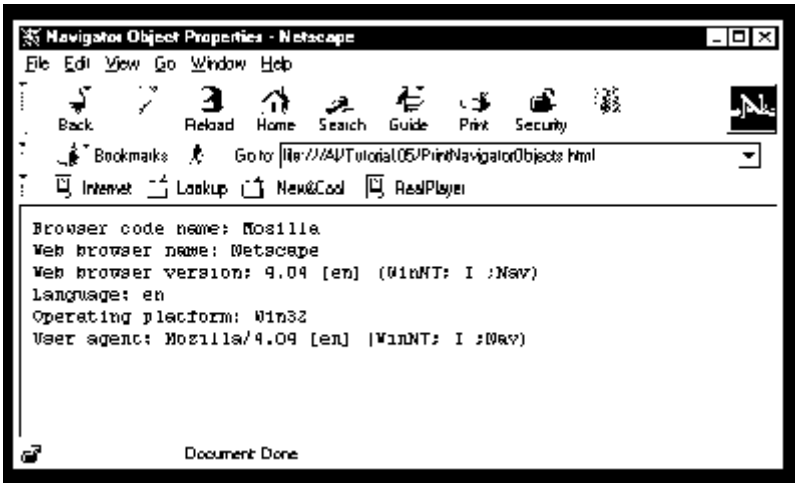


图 5-24　领航员属性程序的输出

下面将打印出使用的浏览器的领航员对象的属性。由于 Netscape Navigator 和 IE 有不同的属性，最好利用 for…in 语句来打印出对象的全部属性。

提示：在第 4 章已经学习过 for…in 语句。

打印使用的浏览器的领航员对象的属性：

1. 启动文本或 HTML 浏览器，创建一个新的文档。

2. 输入文档的开始标签。

```
<HTML>
<HEAD>
<TITLE>Navigator Properties</TITLE>
</HEAD>
```

3. 输入标签<BODY>开始 HTML 文档的主体，并按回车。

4. 添加开始的<PRE>标签和 JavaScript 程序的开始标签。

```
<PRE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

5. 输入下面的 for…in 语句来打印领航员对象的所有属性的名称和取值。

```
for (prop in navigator) {
      document.writeln(prop + ": " + navigator [prop ]);
}
```

6. 添加下面的结束标签：

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

7. 保存并关闭文件到文件夹 Tutorial.05 中，文件名称为 NavigatorObject.html。在浏览器中打开该文件。图 5-25 显示了程序在 Navigator 中的输出结果，图 5-26 显示了程序在 IE 中输出的结果。

8. 关闭浏览器窗口。

帮助：图 5-25 包括了 plugins 属性，该属性是 Navigator 独有的。图 5-26 包括了在 Navigator 中不支持的属性，例如 systemLanguage 和 userLanguage 属性。

5.2.9 帧和窗口

当在多个帧和窗口中工作时，需要能够在 JavaScript 代码中指出单独的帧和窗口。举

例来说，当创建一个新的窗口时，可能需要改变在该窗口中显示的内容。或者在一个窗口中有多个帧，可能需要根据一个帧中的链接改变另一个帧的显示内容。回顾第 1 节，在 JavaScript 对象模型中的一些对象（包括帧对象）实际是对象的数组。帧对象包含了一个包括窗口中所有帧的数组 frames[]。窗口中的第一个帧即数组元素 frame[0]，第二个为 frame[1]，依此类推。如果一个窗口没有包含帧，frames[]数组就是空的。如果要引用在同一个帧集合里的另一个帧，必须将窗口对象的 parent 属性和 frames[]数组结合起来用。举例来说，如果创建一个含有 4 个帧的 HTML 文档，这些帧就可以用 parent.frames[0]、parent.frames[1]、parent.frames[2]和 parent.frames[3]分别来表示。

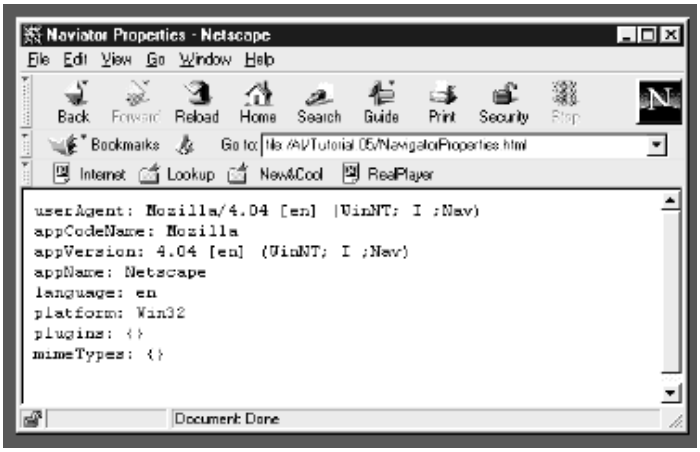


图 5-25　NavigatorObject.html 在 Navigator 中的输出

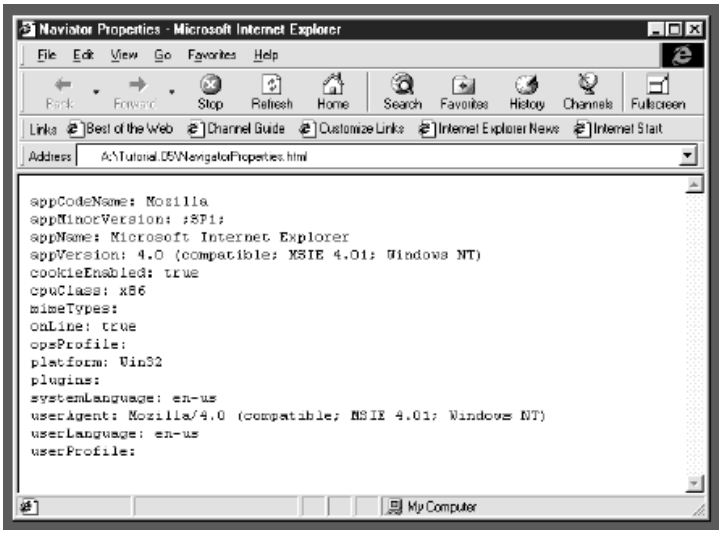


图 5-26　NavigatorObject.html 在 IE 中的输出

为了更好地了解 parent 属性和 frames[]数组，考虑在图 5-27 中显示的 HTML 文档。

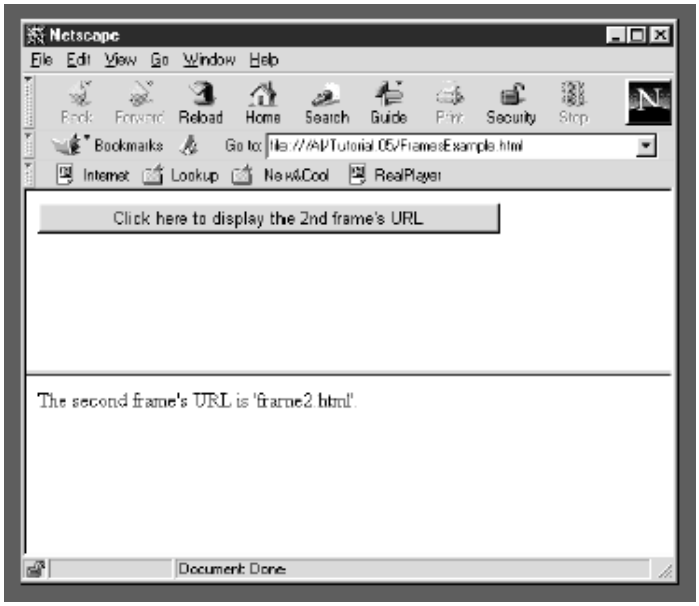


图 5-27 关于 parent 属性和 frames[]数组的例子

图 5-27 中的两个帧由下面的代码创建：

```
<FRAMESET ROWS="50%, 50%">
    <FRAME SRC="frame1.html" NAME="FirstFrame">
    <FRAME SRC="frame2.html" NAME="SecondFrame">
</FRAMESET>
```

第一个帧通过<INPUT>标签的 onClick 事件调用第二帧的 URL 地址，代码如下：

```
<INPUT TYPE="button"
     VALUE="Click here to display the 2nd frame ’s URL "
onClick="alert(parent.frames [1 ].location.href);">
```

正如在代码中看到的，语句 parent.frames [1].location.href 返回第二个帧的 URL 地址。如果让该按钮返回第一帧的 URL，则代码为 parent.frames [0].location.href，因为它是 frames[]数组中的第一个元素。注意，如果需要显示定位对象的 href 属性，必须在语句中包括定位对象，因为必须将对象的所有祖先都列出来（除了窗口对象）。

提示：需要注意，每个帧都有自己的窗口对象，就像有自己的定位、历史和领航员对象一样。

除了用 frames[]数组外，还可以运用在<FRAME>标签中给帧指定的名称来引用它。语句 alert(parent.SecondFrame.location.href);同样标示了第二个帧的 URL。

用 parent 属性来表示一个帧集合中的帧是相当简单直接的。但是当需要用到嵌套的帧集合时，引用一个单独的帧就比较麻烦了。要引用一个比当前帧集合高两层的帧，必须用两个 parent 属性。如果正在用一个嵌套的帧集合工作，并且需要引用包含嵌套的帧的上层帧的 URL，可以利用语句 parent.parent.frames[2].location.href 来实现该功能。图 5-28 显示了图 5-27 程序的一个变体。在该图中，右边的两个帧被嵌套在一个父帧集合里。左边的帧是父帧集合里的第一个帧。右边的两个帧的父帧是其集合中的第二个帧。对于包含了按钮（“获得第一列帧 URL”的按钮）的帧，可以用语句 parent.parent.frames [0].location.href 来引用左边的第一帧。

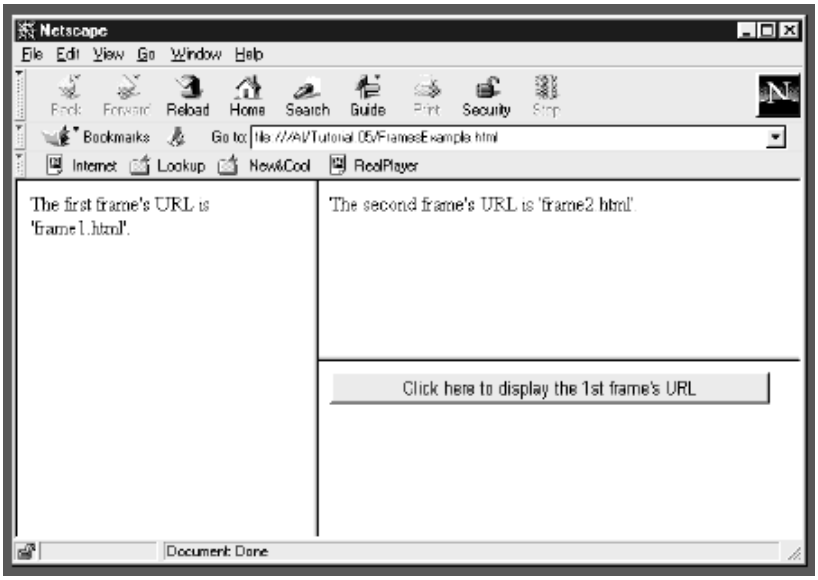


图 5-28　parent.parent 引用的例子

另一个用来引用窗口的属性是窗口对象的 top 属性。该属性引用一个 HTML 文档中最顶层的窗口。当用帧结构工作时，top 属性指的是构建帧集合的窗口。例如，如果创建父帧集合的代码在文件 FrameExample.html 中，那么语句 top.location.href 将返回 FrameExample.html 文档的 URL 地址，无论该语句在哪个帧中使用，返回的结果都是一样的。如果 top 属性用在一个不包含帧的 HTML 文档里，那么它就代表了窗口本身。

5.2.10　总结

- ✧ 帧是一个独立的、可滚动的 Web 浏览器窗口区域，每个帧都可以保留自己的 URL。
- ✧ 每个帧都有自己的窗口对象，与文档中的其他帧区分开。
- ✧ HTML 文档用<FRAMESET>…</FRAMESET>标签对来划分帧。
- ✧ <FRAMESET>标签中的 ROWS 属性决定了一个帧集合中，在水平方向上创建的帧的数目。COLS 属性则决定了在垂直方向上创建的帧的数目。

- ✧ <FRAME>标签用来设定单独的帧的选项，包括帧的 URL 地址。SRC 属性指定了在一个帧中将被打开的 URL 地址。
- ✧ 在帧集合中，URL 按照<FRAME>标签的排列顺序，从左到右，从上至下，依次打开 URL 地址。
- ✧ TARGET 属性指定了 URL 将在哪个帧或窗口中打开。
- ✧ <BASE>标签为一个 HTML 文档中的所有链接指定一个帧或窗口作为默认的目标。
- ✧ 被包括在其他帧里的帧称为嵌套的帧。
- ✧ NORESIZE 属性禁止用户对一个帧尺寸的调整。
- ✧ 可以用 SCROLLING 属性控制一个帧的滚动条。
- ✧ 在 Web 浏览器不支持帧结构时，可以用<NOFRAMES>…</NOFRAMES>标签对向用户显示一条替代信息。
- ✧ 定位对象包含了对浏览器当前窗口中的文档进行操作的若干属性和方法。
- ✧ 定位对象的 reload()方法等同于 Netscape Navigator 的重新载入按钮或 IE 中的重新刷新按钮。
- ✧ 定位对象的 replace()方法用另一个不同的 URL 替代当前载入的 URL 地址。
- ✧ 历史对象包含了在当前 Web 浏览器会话期间，曾经打开过的所有文档的历史记录。
- ✧ 历史对象的 back()和 forward()方法允许程序在浏览器的历史记录中向前或向后移动。
- ✧ 历史对象的 go()方法用来转到一个指定的、曾经浏览过的 Web 页面。
- ✧ 领航员对象用来获得关于当前 Web 浏览器的信息。
- ✧ 帧对象包括了一个 frames[]数组，该数组包含了一个窗口内的所有帧。
- ✧ 如果需要引用同一个帧集合内的帧时，可以将窗口对象的 parent 属性和 frames[]数组结合起来。
- ✧ 当需要引用一个比当前帧集合高一个层次的帧时，需要采用第二个 parent 属性。
- ✧ top 属性指定 HTML 文档中最顶层的窗口。

5.2.11 问题

1. 下面哪个标签用来创建帧？

a. <BEGIN FRAME>...</END FRAME>

b. <FRAMESET>...</FRAMESET>

c. <NEW FRAME>...</NEW FRAME>

d. <FRAMEBUILD>...</FRAMEBUILD>

2. 一个帧的尺寸可以用占屏幕的百分比和_____来确定。

a. inches

b. picas

c. pixels

d. 一个 Web 浏览器的内部度量单位

3. 下面哪个符号用来将屏幕余下的空间分配给一个帧？

a. *

b. &

c. %

d. #

4. 对帧集合中，根据每个帧的<FRAME>标签顺序，如何为每个帧指定 URL？

a. 按照字母顺序

b. 从上到下，从左到右

c. 从左到右从上到下

d. 根据<FRAME>标签的 ORDER 属性

5. 将 MyHomePage.html 载入，下面哪个语句是正确的？

a. <FRAME HREF="MyHomePage.html">

b. <FRAME URL="MyHomePage.html">

c. <FRAME HTML="MyHomePage.html">

d. <FRAME SRC="MyHomePage.html">

6. <A>标签的哪个属性决定了一个 URL 地址要在哪个帧或窗口中打开？

a. OPENINWIN

b. SELECT

c. GOAL

d. TARGET

7. 下面哪个标签用来对一个 HTML 文档中的所有链接指定一个帧或窗口作为默认的目标？

a. <BASE>

b. <SOURCE>

c. <TARGET>

d. <DEFAULT>

8. 一个包含在另一个帧集合里的帧集合，称为_____的帧。

a. 控制

b. 相对

c. 嵌套

d. 内部

9. 为禁止用户调整一个帧的大小，需要在<FRAME>标签里包含哪个属性？

a. NORESIZE

b. RESIZE=NO

c. FIXED

d. LOCKED

10. 下面哪个<FRAME>标签的属性能够将一个单独的帧的滚动条屏蔽掉？

a. verticalscroll=0, horizontalscroll=0

b. showScrollbars=off

c. scrolling=no

d. scrollbars=no

11. 下面哪个<FRAME>标签的属性用来决定一个帧的左右空白？

a. SIDEMARGINS

b. INSIDE and OUTSIDE

c. MARGINLEFT and MARGINRIGHT

d. MARGINWIDTH

12. 哪个标签对用来提示用户浏览器不支持帧结构？

a. <NOFRAMES>...</NOFRAMES>

b. <NOSCRIPT>...</NOSCRIPT>

c. <ALTERNATE>...</ALTERNATE>

d. <MISSINGFRAMES>...</MISSINGFRAMES>

13. 历史对象用来：

a. 决定一个 HTML 文档或 Web 站点是否发生了变化

b. 跟踪用户浏览的一个特殊站点

c. 跟踪用户浏览的一个特殊 HTML 文档

d. 保留了一个用户在当前浏览器窗口会话期间曾经打开过的文档的列表

14. 下面哪个不是历史对象的方法？

a. back()

b. forward()

c. go()

d. next()

15. 一个 Web 页面的完整 URL 地址包含于历史对象的哪个属性中？

a. src

b. href

c. hash

d. url

16. 为了用一个 HTML 文档重写另一个文档，并且用新的 URL 替代在历史记录中的旧的 URL，需要用历史对象的哪个方法？

a. reload()

b. refresh()

c. replace()

d. open()

17. 哪个对象用来获取当前 Web 浏览器的信息？

a. Window

b. Browser

c. Explorer

d. Navigator

5.2.12　练习

1. 创建一个有两个帧的 HTML 文档。一个帧占据屏幕左边的 200 像素，另一个占据余下的屏幕空间。在左边的帧中，创建喜欢的站点的链接列表。当用户单击左边帧中的链接时，对应的站点应该在右边的帧中打开。设置帧为不可调整大小的。保存文件到文件夹 Tutorial. 05 中，文件名称为 Favorites.html。

2. 创建一个有两个帧的 HTML 文档。一个帧占据屏幕顶部 75%的控件，另一个占据 25%的底部空间。在底部帧里，用超链建立导航按钮 Previous、Next 和 Go To。用 onClick 事件、历史对象和定位对象的方法实现每个按钮的功能。保存文件到 Tutorial.05 文件夹中，文件名称为 PersonlNavigator.html。

3. 创建一个 HTML 文档，该文档包含一个按钮，单击按钮可以在一个提示对话框中，显示浏览器的领航员对象的所有属性。在每个属性名称间添加换行符。保存文件到 Tutorial.05 文件夹中，文件名称为 BrowserProperties.html。

第 6 章　表　　单

案例

WebAdventure 的许多客户需要在线的客户登记、说明表格、调查、采购，还要求游戏也作为他们网站的一部分。站点用表单来创建这些应用。WebAdventure 的一个客户，一个新的软件公司，需要建立一个在线的产品注册表来收集客户的信息。现在，WebAdventure 委托您来学习如何建立这个在线的产品注册表。

预览产品注册表

在本章里创建的表单是一个产品注册表。在 6.1 节里，将学习如何将不同的表单标签组合在一起，形成一个产品注册表。在 6.2 节里，将学习如何用 JavaScript 来验证和检查表单的数据，同时将表单的数据传递到 Web 服务器或一个 E-mail 地址。

1．在 Web 浏览器中，打开 Tutorial6_ProductRegistration.html 文件，该文件在文件夹 Tutorial.06 中。Tutorial6_ProductRegistration.html 文件包括两个帧。底部的帧显示产品注册表的第一个页面 Tutorial6_CustomerInfo.html 的信息。在用户单击了 Tutorial6_CustomerInfo.html 文件底部的“ Next ”按钮后，将会显示第二个页面 Tutorial6_ProductInfo.html。页面顶端的帧包括了隐藏的表单域，当用户在两个页面间切换时，这些域保存每个输入域的取值。图 6-1 显示了客户信息表单在 Navigator 中的输出效果。

2．在往表单的输入域输入数据之前，滚动到 Tutorial6_CustomerInfo.html 表单的底端，并且单击“ Next ”按钮。将看到有一个提示对话框，提示尚有若干域没有输入。将要求输入的域添入数值。如果发现有输入错误，请单击“ Reset ”按钮重新对表单进行输入。当完成输入后，请单击“ Next ”按钮显示第二页 Tutorial6_ProductInfo.html，图 6-2 显示了第二个表单。

3．输入产品信息，并且单击“ Submit ”按钮。为了提交表单数据，必须填写序列号和数据输入域，否则将有一个警告对话框提示您。在 Tutorial6_ProductInfo.html 表单中的“ Submit ”按钮用来将表单数据送到一个 Web 服务器。由于并没有可以传送的服务器，单击“ Submit ”按钮将不产生作用。在本章的结尾，将学习如何将表单数据传送到一个 E-mail 地址。

4．完成浏览后，请关闭 Web 浏览器窗口。

5. 接下来，请在文本或 HTML 编辑器中检查 Tutorial6_CustomerInfo.html 和 Tutorial6_ProductInfo.html 的代码。在<BODY>部分里的表单标签创建了在上图中显示的

表单。在<HEAD>部分中的<SCRIPT>…</SCRIPT>标签对中的代码将每个输入域的值拷贝到在 Tutorial6_TopFrame.html 文件顶部的隐藏域中。隐藏域负责当用户在两个页面之间切换时保存每个输入域的值。将隐藏域的值（而不是表单中的输入域的值）传送到服务器。

6．当查看完代码后，请关闭文本或 HTML 编辑器。

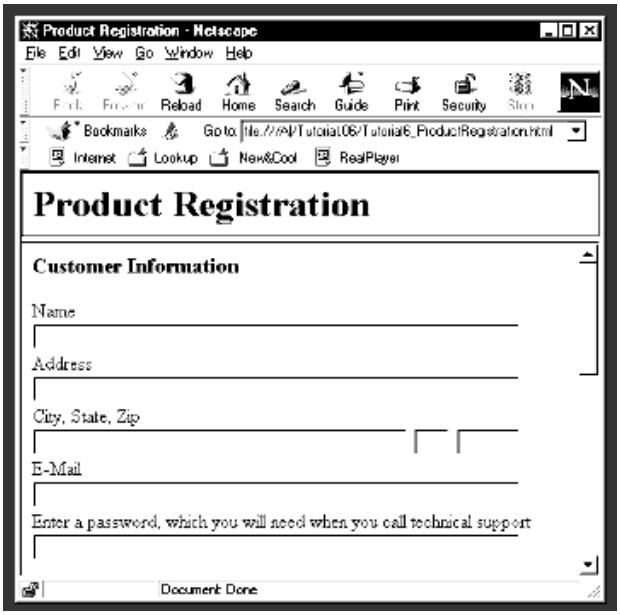


图 6-1　在 Web 浏览器中的 Tutorial6_CustomerInfo.html

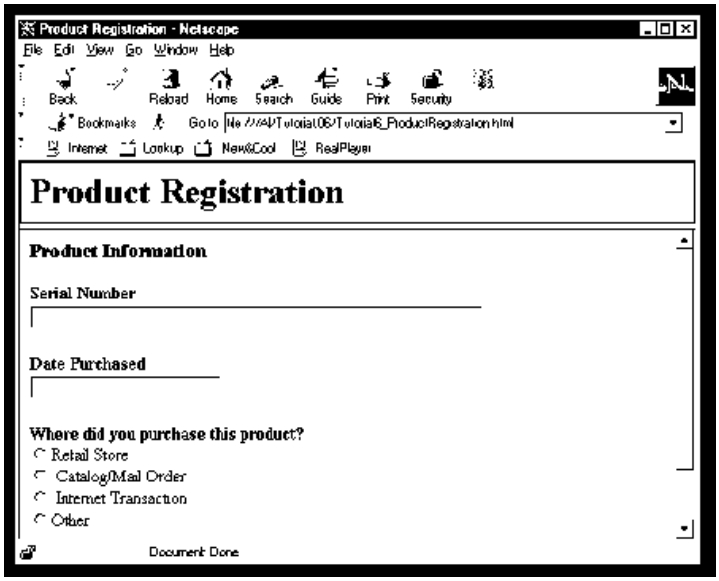


图 6-2　Tutorial6_ProductInfo.html 文件

6.1 在 JavaScript 中使用表单

本节目标

在本节里将学习：

- ✧ 如何使用 HTML 表单
- ✧ 关于通用网关接口
- ✧ 如何使用<FORM>标签
- ✧ 关于表单元素
- ✧ 如何创建和使用输入域
- ✧ 如何创建选择列表域
- ✧ 如何创建多行文本域

6.1.1 表单总览

表单是在 JavaScript 中使用的最多的 HTML 元素之一。许多 Web 站点用表单从用户处收集信息，并将信息传送到服务器去处理。在 Web 中可能遇到的典型表单包括订货单、调查表和申请表。另一种常用的表单是用来收集用户的检索信息的表单。当输入了检索信息后，将信息传送到服务器中的一个数据库。服务器用在搜索表单中收集到的信息检索数据库，然后将结果返回 Web 浏览器。可以用 JavaScript 来检查一个表单的域是否正确输入，或者执行传送到服务器前的其他数据处理。如果没有了 JavaScript，则 HTML 只能够将表单数据送到服务器处理。

要处理从 Web 浏览器传送到服务器的数据，需要用到一个特殊协议，通用网关接口（Common Gateway Interface，CGI）。虽然还有其他的处理方法，例如 ASP、ISPI、NSAPI，但 CGI 是最早和最常用的方法。虽然和 JavaScript 编程没有多少关系，在本章里还是要学习一些 CGI 知识，因为处理服务器端的数据是 Web 编程的一个重要方面。在本章还要学习使用表单和 JavaScript 创建不需要服务器处理数据的程序和 Web 页面。

提示：因为本书的重点是 JavaScript 编程，所以没有涵盖表单的所有知识。如果需要设计和格式化表单的信息，请参看《Creating Web Pages with HTML》（Patrick Carey 著，Course Technology 出版社出版）。

还可以使用 HTML 表单创建许多类型的程序，这些程序不需要往服务器传送数据。例如，在第 2 章中创建的计算器的程序，就包含了一个表单，而且用 JavaScript 命令实现它的功能。此外，表单提供了一种和用户交互的方法，例如在 HTML 文档中的按钮和文本框。

如果没有表单，则除了链接外，几乎没有 HTML 元素可以用来和用户交互。一个 HTML 文档中的表单可以只包含一个按钮，也可以包括许多复杂元素。图 6-3 显示了一个包含了客户导航按钮的表单。

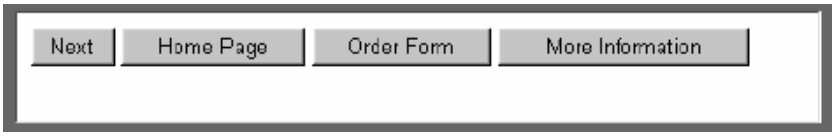


图 6-3　用表单创建的客户导航按钮

在前面的章节里，已经看到了一个常用表单元素：<INPUT>标签，在几个例子和练习中已经使用过。还使用过<FORM>标签。在第 2 章里，学习了 JavaScript 事件，并了解到<INPUT>标签有关的事件是最常用的 JavaScript 事件。虽然学习过表单的一些知识，但是还有很多表单的其他知识需要学习。

6.1.2　通用网关接口

通用网关接口（Common Gateway Interface，CGI），是一种允许 Web 页面和服务器端的程序通信的简单协议。CGI 的功能是启动服务器端的程序，然后和程序间进行环境变量的传送和接收。环境变量是操作系统的一部分，它和 JavaScript 变量不同，后者只是函数或程序的一部分。一个 CGI 环境变量的例子是 server_name，它指定了一个 Web 服务器的域名或 IP 地址。处理 CGI 环境变量的 Web 服务器端程序称为 CGI 脚本或 CGI 程序。CGI 程序可以执行多种功能。不要将 CGI 程序和 CGI 协议本身混淆。CGI 协议的基本目的是将页面中接收的数据送到服务器端的程序，然后将程序的回应送回 Web 页面。运行在服务器上的程序可以是一个数据库程序或其他类型的客户应用程序。可以用一种脚本语言创建服务器端程序，例如 AppleScript、Perl 和 TLC。也可以用编程语言创建服务器程序，例如 Visual Basic 和 Visual C++等。

提示：CGI 脚本常被放在一个 Web 服务器中的 bin 或 cgi-bin 目录。CGI 脚本文件通常以 .cgi 为后缀。如果看到一个 URL 地址 http://www.ExampleWebPage/cgi-bin/example_script.cgi，那么可以确定，这是一个 CGI 脚本程序。

CGI 脚本用来处理输入到 HTML 表单中的信息。HTML 表单元素用来将输入的信息传送到 CGI 环境变量中。然后，CGI 环境变量被传送到服务器上的一个 CGI 脚本。CGI 脚本执行诸如查询等动作，然后将结果送回请求的 Web 页面或创建一个新的 HTML 文档。图 6-4 中显示了一个用 Perl 语言写的 CGI 脚本，该脚本产生一个新的 Web 页面作为请求页面的回应。

```
# !/usr/local/bin/perl
#
# formdata.pl — "Form Data Received" program
#
# The following line prints the CGI response header that
# is required for HTML output. Each    \n sends
# a blank line (response headers must be followed
# by a blank line).
print "Content-type: text/html \n \n" ;
# The following lines print the HTML response
# page to STDOUT:
print <<EOF ;
<HTML>
<HEAD><TITLE>Form Data Received</TITLE></HEAD>
<BODY>
<H1>Your form data has been received.</H1>
</BODY>
</HTML>
EOF
exit ;
```

图 6-4　用 Perl 语言写的 CGI 脚本

不必担心理解不了脚本中的 Perl 语言。这个例子只是为了说明 CGI 允许与 HTML 页面和 Web 服务器应用或者除了 JavaScript 外的其他语言进行交互。

提示：本书的目的是传授 JavaScript 语言，并不解释创建 CGI 脚本的其他编程语言的结构或语法。当本书的例子中包含 CGI 程序时，只解释 CGI 程序的功能，并不关心脚本或程序是如何工作的。

6.1.3　<FORM>标签

所有的表单都用<FORM>…</FORM>标签对创建。<FORM>…</FORM>标签对在 HTML 文档中创建一个表单，其中包含了创建表单的所有文本和标签。在一个 HTML 文档中，可以包含需要的任意多的表单。但是，与帧结构不同，不能够将一个表单嵌套在另一个表单中。要保证用</FORM>标签结束每个表单。如果 JavaScript 解释器在一个</FORM>前遇到一个新的<FORM>标签，那么在第二个表单标签后面的所有表单元素都将添加到第一个表单中。图 6-5 显示了一个包含两个表单的 HTML 文档。图 6-6 显示了文档在 Web 浏览器中的输出。

```
<HTML>
<HEAD>
<TITLE>Two Forms</TITLE>
</HEAD>
<BODY>
<H1>This document contains two forms.</H1><P>
<H2>This is the first form.</H2><P>
<FORM>
<INPUT TYPE="text">
<INPUT TYPE="button", VALUE="Click Me">
</FORM>
<HR>
<H2>This is the second form.</H2><P>
<FORM>
<INPUT TYPE="text">
<INPUT TYPE="button", VALUE="Click Me"
</FORM>
</BODY>
</HTML>
```

图 6-5　包含两个表单的 HTML 文档

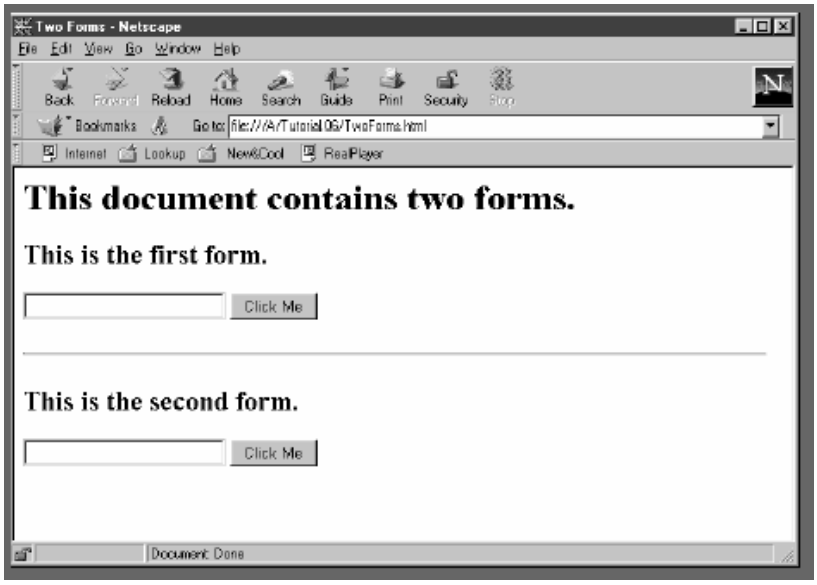


图 6-6　包含两个表单的 HTML 文档的输出

<FORM>标签包括若干可以使用的属性，列在表 6-1 中。

表 6-1 <FORM>标签的属性

属 性	描 述
ACTION	指定一个表单的数据将被提交到的目标 URL。如果包括了该属性，则数据将被传送到包含该表单的 URL。通常该属性指定为一个服务器端程序的 URL 或一个 E-mail 地址
METHOD	决定一个表单的数据以何种方式提交。该属性的两个选项是 GET 和 POST。GET 为默认选项，将一个表单的数据作为一个长字符串，传送到 ACTION 属性指定的 URL。POST 选项将表单数据分隔传输。虽然 GET 是默认选项，但推荐使用 POST 选项，因为它允许服务器从 URL 接受分隔开的数据
ENCTYPE	设置被传送数据的格式。默认的值为 application/x-www-form-urlencoded
TARGET	指定一个窗口，在该窗口中显示从服务器返回的所有结果
NAME	为表单指定一个名称

ENCTYPE 属性指定了一个编码协议，称为多用途的网际邮件扩充协议（Multipurpose Internet Mail Extension）或 MIME。用 MIME 编码保证了数据在 Internet 上传输时不被中断。开发 MIME 协议最初是为了允许不同的文件类型作为 E-mail 的附件进行传输。现在，MIME 已经成为了在 Internet 上交换文件的一种标准方法，虽然到目前为止，该技术还在发展。MIME 类型由一个被左斜杠“ / ”分成两部分的代码表示。第一部分标识了 MIME 类型，第二部分标识了 MIME 子类型。默认的 MIME 类型 application/x-www-form-urlencoded 指定了一个表单的数据应该按照一个长字符串进行编码。在 ENCTYPE 属性可以采用的另两个的 MIME 类型是 multipart/form-data，该类型将每个域作为单独的部分编码，以及 text/plain 用来将表单数据提交到一个 E-mail 地址。

提示：如果需要更多的关于编码的信息，请查看位于 http://www.ics.uci.edu/ pub/ietf /html/rfc1867.txt 中的文档。

考虑一下在图 6-7 和图 6-8 中的代码，这些代码显示了如何使用<FORM>标签的属性。图 6-7 中的代码创建了一个包含三个帧的文档，其中的一个帧包含了一个表单。图 6-8 中的代码创建了包含在帧 subscription 中的简单表单。图 6-9 显示了程序在 Web 浏览器中的输出。

在图 6-8 中的<FORM>标签包含了一个 ACTION 属性，将表单的数据提交到 URL 地址 http://www.Emily-the-Chimp/cgi-bin/subscribe 中。METHOD 属性指定了表单的数据应当以 POST 方法提交，而不是用默认的 GET 方法。因为 ENCTYPE 属性被省略，表单数据将按照默认的 application/x-www-form-urlencoded 格式编码。表单还被赋给了一个名称：subscriptionForm。<FORM>标签的最后一个属性 TARGET 设置为 dialog。TARGET 属性指定了从服务器返回的信息应该在哪个窗口中显示。在该例中，TARGET 属性指定了图 6-7 中创建的帧 dialog 作为目标窗口。用 Subscribe 按钮提交表单数据。当服务器接收到数据后，虚构的 CGI 脚本 subscribe 将该 E-mail 地址添加到数据库中，然后返回一个消息到 dialog

帧中，返回结果如图 6-10 所示。

```
<HTML>
<HEAD>
<TITLE>Emily the Chimp</TITLE>
</HEAD>
<FRAMESET ROWS="85%,*">
<FRAMESET COLS="52%,*">
<FRAME SRC="Emily.jpg" NAME="Emily">
<FRAME SRC="dialog.html" NAME="dialog">
</FRAMESET>
<FRAME  SRC="subscription.html"
NAME="subscription">
</FRAMESET>
</HTML>
```

图 6-7　包含三个帧的 HTML 文档

```
<HTML>
<BODY>
<FORM ACTION="http://www.Emily-the-Chimp/cgi-bin/subscribe"  METHOD="post"
NAME="subscriptionForm" TARGET="dialog">
<INPUT TYPE="text" SIZE=50>
<INPUT TYPE="submit" VALUE="Subscribe">
</FORM>
</BODY>
</HTML>
```

图 6-8　预订帧

图 6-9　黑猩猩 Emily 页面

图 6-10　提交表单后的黑猩猩 Emily 页面

下面将创建在本章开头看到的产品注册程序。用帧结构创建产品注册文档。首先创建载入帧集合的主 HTML 文档，然后为每个帧创建 HTML 文档。

创建主 HTML 文档，它载入用于产品注册表的帧集合：

1. 启动文本和 HTML 编辑器，创建一个新的文档。

2. 输入文档的开始标签。

```
<HTML>
<HEAD>
<TITLE>Product Registration</TITLE>
</HEAD>
```

3. 添加<FRAMESET ROWS="60,*">来启动帧集合。顶层的帧设置为 60 像素，它包含了产品登记表的标题。底层的帧占据余下的屏幕空间，包含了表格本身。

4. 添加下面的两个<FRAME>标签。第一个标签打开一个名为 TopFrame.html 的 HTML 文件，该文件包含了表格的标题。第二个标签打开一个名为 CustomerInfo.html 的 HTML 文件，该文件包含了第一个表单。包含 TopFrame.html 的窗口命名为 topframe，底层的窗口命名为 bottomframe。

```
<FRAME SRC="TopFrame.html"NAME="topframe"SCROLLING=no>
<FRAME SRC="CustomerInfo.html" NAME="bottomframe">
```

5. 输入结束标签。

```
</FRAMESET>
</HTML>
```

6. 保存文件到文件夹 Tutorial.06，文件名称为 ProductRegistration.html。

7. 关闭文件 ProductRegistration.html。

下面创建 TopFrame.html 文件，该文件显示在名为 topframe 的帧中：

1．启动文本和 HTML 编辑器，创建一个新的文档。

2．输入下面的标签，创建包含一个头标的简单 HTML 文件。

```
<HTML>
<BODY>
<H1>Product Registration</H1>
</BODY>
</HTML>
```

3．保存文件到文件夹 Tutorial.06，文件名称为 TopFrame.html。

4．关闭文件。

最后创建 CustomerInfo.html 文件，该文件包含产品注册表的表单代码：

1．启动文本和 HTML 编辑器，创建一个新的文档。

2．输入开始的标签。

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

3．输入<H3>Customer Information</H3>作为产品登记表的子标题。

4．添加下面两个标签，创建表单部分。在该部分，将添加表单元素。

```
<FORM NAME="CustomerInfo">
</FORM>
```

提示：由于实际上并不将该表单提交到一个 CGI 程序，所以该<FORM>标签并未包含 ACTION、METHOD 和 ENCTYPE 属性。

5．添加下面的结束标签：

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

6．保存文件到文件夹 Tutorial.06，文件名称为 CustomerInfo.html。请先不要打开该文件，因为表单中还未包含任何元素。

7．关闭文本或 HTML 编辑器。

6.1.4 表单元素总览

在<FORM>…</FORM>标签对内，有三种标签可以用来创建表单元素：<INPUT>、<SELECT>和<TEXTAREA>。正如您所知道的，<INPUT>标签用来创建一个和用户交互的输入域。<SELECT>标签用来显示一个下拉菜单或滚动列表形式的选择列表。<TEXTAREA>标签用来创建一个用户可以输入多行文本的文本输入框。任何一个用户可以输入数据，进行选择或改变的表单元素，我们称为域。

<INPUT>、<SELECT>和<TEXTAREA>标签可以包含名称 NAME 属性，取值 VALUE 属性。NAME 属性给标签定义了一个名称，VALUE 属性则给标签定义一个默认的取值。当往 CGI 脚本传输表单数据时，表单的数据是按照 name=value 标签对的顺序提交的。举例来说，有一个用语句<INPUT TYPE="text" NAME="company_info" VALUE="ABC Corp."> 创建的文本输入域，那么一个 name=value 的标签对 company_info=ABC Corp，将被传送到 CGI 脚本（除非默认值因为某些原因改变了）。如果需要将表单传送到 CGI 脚本，则必须为每个<INPUT>、<SELECT>和<TEXTAREA>标签指定一个名称。在表单的数据被提交之前，可以不用为一个域指定默认值或输入取值，因为一个空值或 null 的取值是可以被接受的。但是，在数据提交到 CGI 脚本之前，用 JavaScript 对数据进行校验是一个很好的主意。如果需要用 JavaScript 校验数据，需要对 VALUE 属性指定一个取值。

提示：将在第 2 节里学习如何对表单的数据进行校验。

6.1.5 输入域

<INPUT>标签用来创建不同类型的、用于收集信息的交互表单元素——输入域。<INPUT>标签的属性包括 ALIGN、CHECKED、MAXLENGTH、NAME、SIZE、TYPE、VALUE 和 SRC。TYPE 属性决定了元素的类型，是一个必须的属性。TYPE 属性的有效取值包括文本域、密码域、单选按钮、复选按钮、复位按钮、常用按钮、提交按钮、图片和隐藏域。每个属性的取值必须用引号引起来，以便于在 Navigator 和 IE 中都能正确地运行。表 6-2 列出了<INPUT>标签的属性，并描述了它们的功能和语法。

文本框

TYPE 属性为 text 的<INPUT>标签创建一个接收单行文本的简单文本框。在<INPUT TYPE ="text">标签中，可以包含 NAME、VALUE、MAXLENGTH 和 SIZE 属性。下面的标签创建了图 6-11 显示的文本框。

表 6-2　<INPUT>标签的常用属性

属　　性	描　　述
ALIGN	指定用 TYPE 属性创建的一个图片的对齐方式。有效的取值包括 ABSBOTTOM、ABSMIDDLE、BASELINE、BOTTOM、LEFT、MIDDLE、RIGHT、TEXTTOP 和 TOP
CHECKED	指定一个复选或单选按钮是否被选中
MAXLENGTH	设置一个文本域可以接受的最大字符数
NAME	为表单元素指定一个名称，用在 name=value 标签对中传输数据
SIZE	指定一个整数，决定一个文本域有多少个字符宽度
SRC	指定一个图片的 URL
TYPE	指定元素的类型；TYPE 属性是必需的。有效的 TYPE 属性包括 text、password、radio、checkbox、reset、button、submit、image 和 hidden
VALUE	设置一个文本域或按钮标签的初始值，用在 name=value 标签对中传输数据

```
<FORM ACTION="http://example_url/cgi-bin/cgi_program"
      METHOD="post" NAME="exampleForm">
Name<BR>
<INPUT TYPE="text" NAME="name"
      VALUE="The White House" SIZE=50><BR>
Address<BR>
<INPUT TYPE="text" NAME="address"
      VALUE="1600 Pennsylvania Ave." SIZE=50><BR>
City, State, Zip<BR>
<INPUT TYPE="text" NAME="city"
      VALUE="Washington" SIZE=38>
<INPUT TYPE="text" NAME="state"
      VALUE="DC" SIZE=2 MAXLENGTH=2>
<INPUT TYPE="text" NAME="zip"
      VALUE="20500" SIZE=5 MAXLENGTH=5>
</FORM>
```

当在文本<INPUT>标签中包含 VALUE 属性时，该属性指定的文本作为表单刚载入时文本框的默认取值，正如图 6-11 中显示的那样。

下面将往 CustomerInfo.html 文件中添加前面的几个文本输入域的标签：

1．在文本或 HTML 编辑器中打开 CustomerInfo.html 文件。

2．在<FORM>…</FORM>标签对内，添加下面的文本输入域标签，它们收集客户的名称、地址、城市、州、邮编和 E-mail 地址等信息。

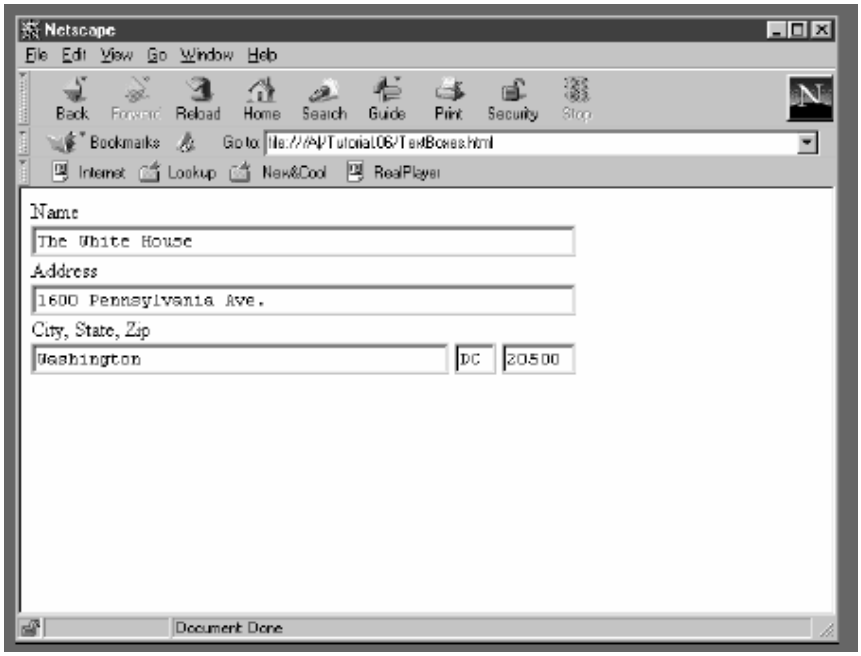


图 6-11 若干<INPUT>标签的输出

```
Name<BR>
<INPUT TYPE="text" NAME="name" SIZE=50><BR>
Address<BR>
<INPUT TYPE="text" NAME="address" SIZE=50><BR>
City, State, Zip<BR>
<INPUT TYPE="text" NAME="city" SIZE=38>
<INPUT TYPE="text" NAME="state" SIZE=2 MAXLENGTH=2>
<INPUT TYPE="text" NAME="zip" SIZE=5 MAXLENGTH=5><BR>
E-Mail<BR>
<INPUT TYPE="text" NAME="email" SIZE=50><BR>
```

3. 保存并关闭 CustomerInfo.html 文档，然后在 Web 浏览器中打开 Product-Registration.html 文件。文件中的文本输入域的显示应当和图 6-12 相似。

4．关闭浏览器窗口。

密码输入框

TYPE 取值为 password 的<INPUT>标签创建一个密码输入域。密码输入域和文本输入域类似。但是，在密码输入域中输入的每个字符都是以星号的形式显示，以免有人从用户的后面看到用户输入的密码。在密码输入域中，可以包含 NAME、VALUE、MAXLENGTH、和 SIZE 属性。下面的代码就创建了一个有 8 个字符长度的密码输入框。

```
<FORM ACTION="http://exampleurl/cgi-bin/cgi_program"
      METHOD="post" NAME="exampleForm">
Please enter a password of 8 characters or less:<BR>
<INPUT TYPE="password" NAME="password" MAXLENGTH=8>
</FORM>
```

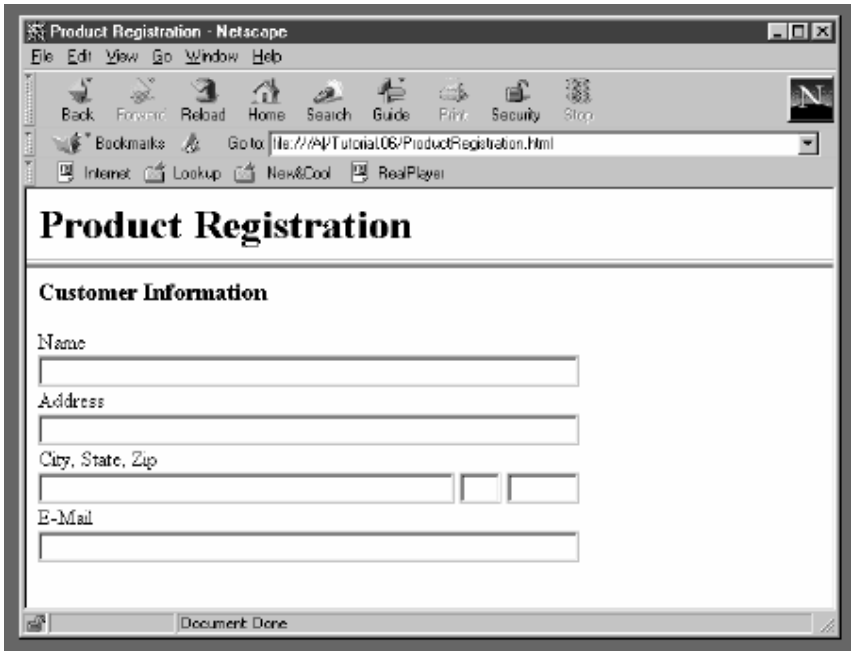


图 6-12　添加了文本输入域的产品登记表程序

下面将往 CutstomerInfo.html 文件中添加密码输入<INPUT>标签，提示用户当需要技术支持时输入密码：

1．在文本或 HTML 编辑器中打开 CustomerInfo.html 文件。

2．在最后一个文本<INPUT>标签后，添加下面的密码输入域的标签，提示用户当需要技术支持时输入密码。

```
Enter a password,which you will need when you call
technical support<BR>
<INPUT TYPE="password" NAME="password" SIZE=50><P>
```

3．保存并关闭 CustomerInfo.html 文档，然后在 Web 浏览器中打开 Product-Registration.html 文件。测试密码输入域，看输入的字符是否都是以星号显示。文件的输出应当如图 6-13 所示。

4．关闭 Web 浏览器窗口。

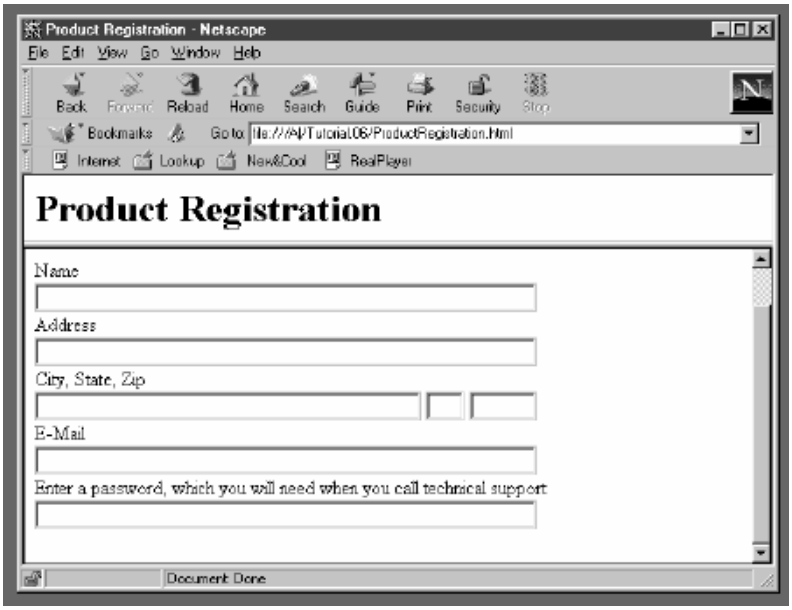


图 6-13 添加了密码输入框的产品注册程序

单选按钮

TYPE 属性为 radio 的<INPUT>标签通常用来创建一组单选按钮。在该组选项中，只能选择一个取值。如果需要创建一组单选按钮，组中的所有按钮必须包含相同的 NAME 属性。每个单选按钮都需要一个 VALUE 属性。当往 CGI 脚本传送表单时，在这一组按钮中只有选中的按钮传送一个 name=value 对。还可以在单选按钮的<INPUT>标签中包括 CHECKED 属性，该属性用来在一组按钮中设定一个初始的选中按钮。如果 CHECKED 属性没有被包括在<INPUT TYPE="radio">标签的按钮组中的任何一个按钮中，则该组中的第一个按钮会被默认的选中。下面的代码创建一个有 5 个单选按钮的按钮组。

```
<FORM ACTION="http://exampleurl/cgi-bin/cgi_program"
METHOD="post" NAME="exampleForm">
Please select your favorite type of music:<BR>
<INPUT TYPE="radio" NAME="music"
      VALUE="jazz">Jazz<BR>
<INPUT TYPE="radio" NAME="music"
      VALUE="classical">Classical<BR>
<INPUT TYPE="radio" NAME="music"
      VALUE="country">Country<BR>
<INPUT TYPE="radio" NAME="music"
      VALUE="rock" CHECKED>Rock<BR>
<INPUT TYPE="radio" NAME="music"
      VALUE="r&b">Rhythm and Blues<BR>
```

```
</FORM>
```

添加单选按钮<INPUT>标签到 CustomerInfo.html 文件中：

1．在文本或 HTML 编辑器中打开 CustomerInfo.html 文件。

2．在密码输入域<INPUT>标签后面，添加下面的单选按钮<INPUT>标签。用户可以选择一个单选按钮来指明他们使用的计算机平台的类型。需要注意的是，每个单选按钮都必须有同样的 NAME 属性 platform。

```
What platform do you use?<BR>
<INPUT TYPE="radio" NAME="platform"
        ALUE="win95-98">Windows 95/98
<INPUT TYPE="radio" NAME="platform"
        ALUE="winnt">Windows NT
<INPUT TYPE="radio" NAME="platform"
        ALUE="unix">UNIX
<INPUT TYPE="radio" NAME="platform"
        ALUE="mac">Macintosh<P>
```

3．保存并关闭 CustomerInfo.html 文档，然后在 Web 浏览器中打开 Product-Registration.html 文件。单选按钮输入域的显示如图 6-14 所示。

4．关闭 Web 浏览器窗口。

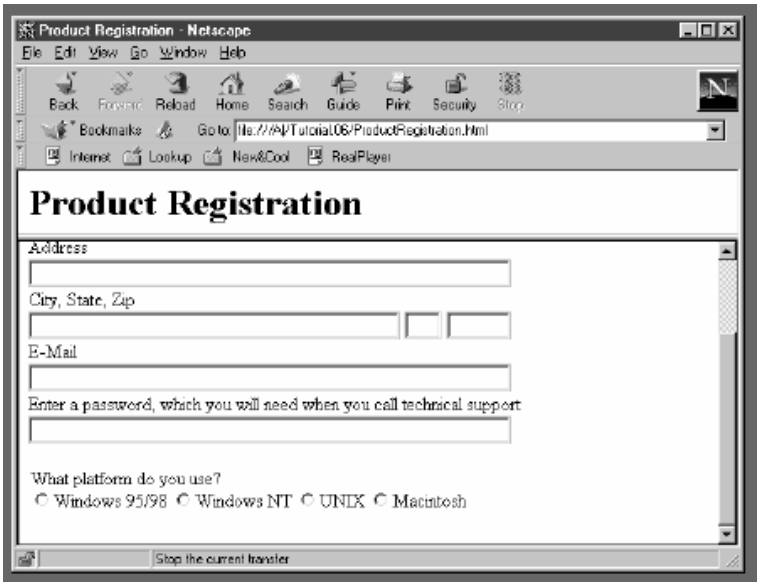


图 6-14　添加了单选按钮输入域的产品注册程序

复选框

TYPE 属性为 checkbox（<INPUT TYPE="checkbox">）创建一个可以被设置为 yes（选

中）或者 no（未选中）的复选按钮。当需要让用户对一个项目进行确定，选择是或不是的情况下，或者允许用户在项目列表中进行多项选择时可以利用复选框。在复选框的<INPUT>标签中包括 CHECKED 属性，将一个复选框的初始取值设为 yes。还可以在复选框的<INPUT>标签中包含 NAME 和 VALUE 属性。当一个复选框被选中后，在表单提交时，复选框的 name=value 字符串就会包含在表单的数据之中。如果复选框未被选中，则在表单提交的数据中将会不包含 name=value 字符串。

下面的代码创建了若干复选按钮：

```
<FORM ACTION="http://example_url/cgi-bin/cgi_program"
      METHOD="post" NAME="exampleForm">
<H3>Which programming languages do you know?</H3>
<INPUT TYPE="checkbox" NAME="prog_languages"
      VALUE="JavaScript"
      CHECKED>JavaScript<BR>
<INPUT TYPE="checkbox" NAME="prog_languages"
      VALUE="Java">Java<BR>
<INPUT TYPE="checkbox" NAME="prog_languages"
      VALUE="Visual Basic">
Visual Basic<BR>
<INPUT TYPE="checkbox" NAME="prog_languages"
      VALUE="Visual C++">
Visual C++<BR>
</FORM>
```

和单选按钮一样，也可以通过给若干复选按钮指定同样的 NAME 属性将其归为一组，虽然每个复选按钮的取值可能不一样。与单选按钮不同，用户可以在一组复选按钮中任意选中按钮，选中的按钮数目不受限制。当在一个表单中有同样名称的多个复选按钮被选中时，就会有多个 name=value 对，每个都用同样的名称提交给 CGI 脚本。在前面的例子中，如果 JavaScript 和 Java 复选框被选中，则两个 name=value 对 prog_languages =JavaScript 和 prog_languages=Java 将会被提交到 CGI 脚本。注意，并不是必须将若干复选按钮用同样的 NAME 属性归为一组。虽然一组复选按钮有相同的名称有助于确定管理复选按钮组，但是通常每个按钮有单独的名称更容易追踪其单独的取值。

下面，将添加复选按钮<INPUT>标签到 CustomerInfo.html 文件，以提示用户选择他们使用的软件类型。在本章的后面，将会拷贝每个复选框到另一个表单中对应的隐藏域中。在拷贝时，为了可以更方便地引用每个复选框，每个复选按钮都被指定了不同 NAME 值。

添加复选按钮<INPUT>标签到 CustomerInfo.html 文件：

1．在文本或 HTML 编辑器中打开 CustomerInfo.html 文件。

2．在最后一个单选按钮<INPUT>标签后面，添加下面的复选框<INPUT>标签，提示

用户输入他们使用的软件的类型。

```
What types of software do you use? (check all
      that apply)<BR>
<INPUT TYPE="checkbox" NAME="wp" VALUE="wp">
      Word Processing<BR>
<INPUT TYPE="checkbox" NAME="ss" VALUE="ss">
      Spreadsheets<BR>
<INPUT TYPE="checkbox" NAME="db" VALUE="db">
      Database<BR>
<INPUT TYPE="checkbox" NAME="gr" VALUE="gr">
        Graphics/CAD<BR>
<INPUT TYPE="checkbox" NAME="pr" VALUE="pr">
      Programming<P>
```

3．保存并关闭 CustomerInfo.html 文档，然后在 Web 浏览器中打开 Product-Registration.html 文件。文件中的复选按钮<INPUT>标签的输出结果如图 6-15 所示。

4．关闭 Web 浏览器窗口。

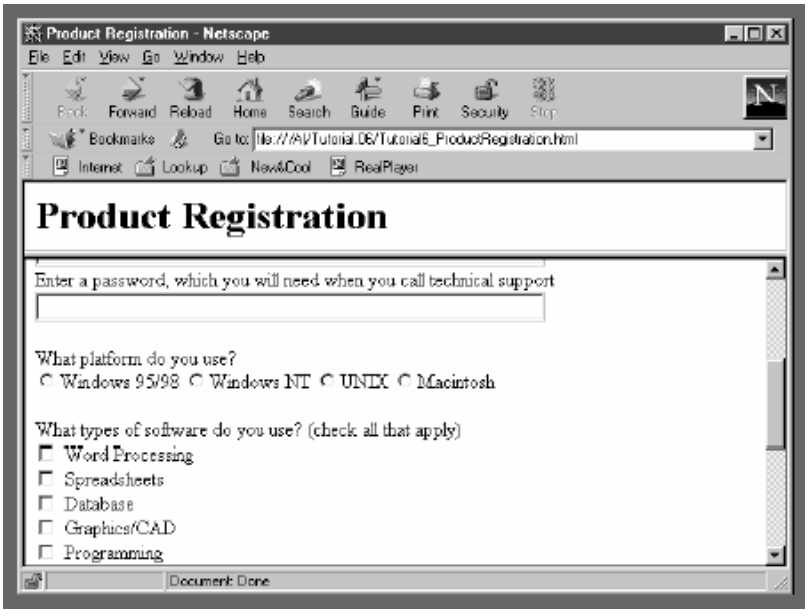


图 6-15　添加了复选框<INPUT>域的产品注册程序

复位按钮

TYPE 属性为 reset（<INPUT TYPE="reset">）的复位按钮将清空表单的所有项目，并且将每个表单的元素设置为其由 VALUE 属性指定的初始值。虽然可以为复位按钮指定一个 NAME 属性，同样可以在 JavaScript 代码中引用该名称，但是没有必要这样做，因为复

位按钮并不作为表单的数据提交到 CGI 程序。如果没有为复位按钮指定 VALUE 属性，则按钮标签显示默认的 reset 值。

下面的代码创建了一个有复位按钮的表单。图 6-16 显示了输入部分数据后的结果页面。

```
<H3>BillingInformation</H3>
<FORM ACTION="http://exampleurl/cgi-bin/cgi_program"
      METHOD="post" NAME="exampleForm">
<B>Name</B><BR>
<INPUT TYPE="text" NAME="name" SIZE=50><BR>
<B>Address</B><BR>
<INPUT TYPE="text" NAME="address" SIZE=50><BR>
<B>City, State, Zip</B><BR>
<INPUT TYPE="text" NAME="city" SIZE=38>
<INPUT TYPE="text" NAME="state" SIZE=2>
<INPUT TYPE="text" NAME="zip" SIZE=5 MAXLENGTH=5><BR>
<B>Credit Card</B><BR>
<INPUT TYPE="radio" NAME="creditcard" CHECKED>VISA
<INPUT TYPE="radio" NAME="creditcard">MasterCard
<INPUT TYPE="radio" NAME="creditcard">
      American Express<BR>
<INPUT TYPE="radio" NAME="creditcard">Discover
<INPUT TYPE="radio" NAME="creditcard">
      Diners Club<BR>
<B>Credit Card Number</B><BR>
<INPUT TYPE="text" NAME="cc#"
      VALUE="xxxx xxxxxx xxxxx" SIZE=50><BR>
<B>Expiration Date</B><BR>
<INPUT TYPE="text" NAME="expdate"
      VALUE="1/1/99" SIZE=50><P>
<INPUT TYPE="reset">
</FORM>
```

如果单击图 6-16 表单中的复位按钮，则每个输入域的内容将被清空并设置为默认值，如图 6-17 所示。

提示：用复位按钮<INPUT>标签创建的按钮的宽度取决于在 VALUE 属性中输入的字符的数目。

下面将添加复位按钮到 CustomerInfo.html 文件：

1．在文本或 HTML 编辑器中打开 CustomerInfo.html 文件。

2．在最后一个复选按钮<INPUT>标签后面，添加<INPUT TYPE="reset">。

3. 保存并关闭 CustomerInfo.html 文档，然后在 Web 浏览器中打开 Product-

Registration.html 文件。文件的输出应当如图 6-18 所示。

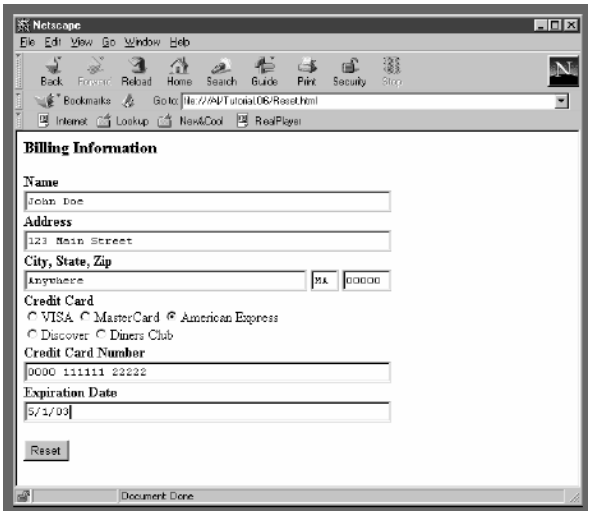


图 6-16　有复位按钮的一个表单

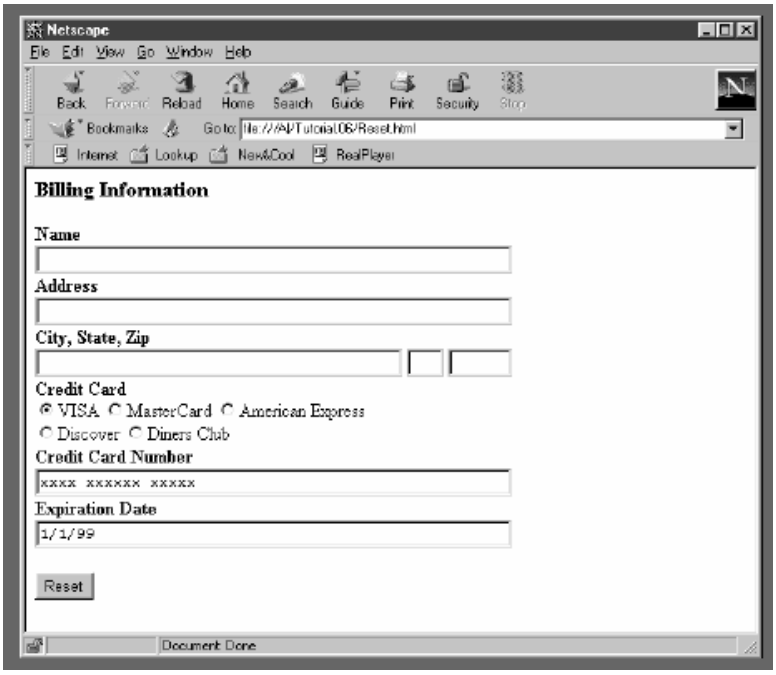


图 6-17　单击复位按钮后的表单输出

4．关闭 Web 浏览器窗口。

命令按钮

TYPE 属性为 button 的<INPUT>标签（<INPUT TYPE="button">）创建一个与在对话框中看到的“ OK ”和“ Cancel ”按钮相似的命令按钮。命令按钮与提交和复位按钮也十分相

似。但是，命令按钮并不像提交按钮那样传送数据到 CGI 脚本，也不像复位按钮那样重置表单的各个域。相反地，命令按钮用一个 onClick 事件的处理程序执行 JavaScript 代码，执行类似于计算等功能。虽然命令按钮并不一定需要事件处理代码，但是它的主要目的是执行 JavaScript 代码。也没有必要在命令按钮中包括 NAME 和 VALUE 属性，因为用户并不能够改变命令按钮的取值。如果包含了这两个属性，那么由 VALUE 属性指定的默认值将和余下的表单数据传送到 CGI 脚本程序。下面的代码创建一个简单的命令按钮。

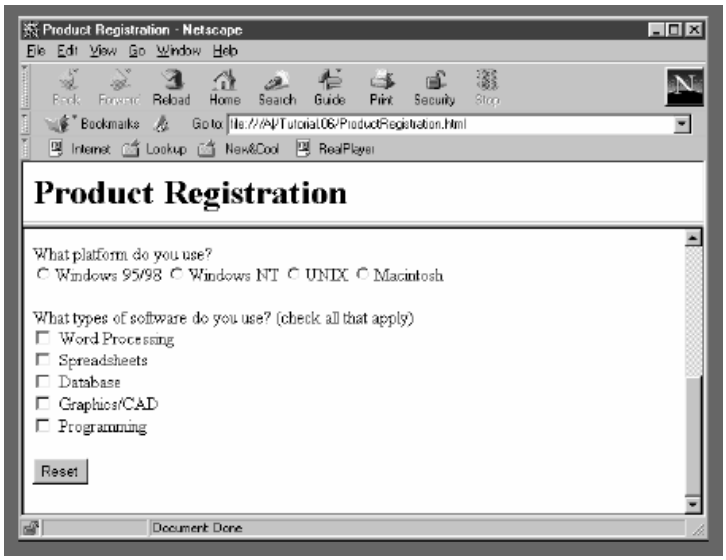


图 6-18 添加了复位按钮的产品注册程序

```
<INPUT TYPE="button" NAME="command_button"
      VALUE="Click Here"
onClick="alert('You Clicked a Command Button');">
```

上述代码创建了一个取值为 Click Here，名称为 command_button 的按钮。在上例中，<INPUT>标签的 onClick 事件处理程序中，显示了一个包含文本 You Clicked a Command Button 的警告对话框。

提示：用<INPUT TYPE="button">创建的按钮的宽度取决于按钮的 VALUE 属性中字符的数目。

下面将添加一个命令按钮到 CustomerInfo.html 文件中。该按钮将在 onClick 事件的处理程序中打开产品注册表的第二个页面。第二个页面是名为 ProductInfo.html 的 HTML 文档。

添加一个命令按钮到 CustomerInfo.html 文件打开产品注册表的第二个页面：

1．在文本或 HTML 编辑器中打开 CustomerInfo.html 文件。

2．在</HEAD>标签前面，添加下面的<SCRIPT>…</SCRIPT>标签对，其中是一个函

数，该函数用定位对象的 href 属性，用 ProductInfo.html 替代 CustomerInfo.html 文件。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
function nextForm() {
        location.href="ProductInfo.html";
}
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
```

3．在 HTML 文档体的复位<INPUT>标签后面，添加下面的代码创建一个名为“ Next ”的按钮。该按钮的 onClick 事件调用在第二步中添加的 nextForm()函数。

```
<INPUT TYPE="button" NAME="next" VALUE=" Next"
          onClick="nextForm()"><P>
```

4.　保存并关闭 CustomerInfo.html 文档。

下面，将创建用户单击“ Next ”按钮时打开的 ProductInfo.html 文件。

创建 ProductInfo.html 文件：

1．打开文本或 HTML 编辑器创建一个新文档。

2．输入文档开始的标签。

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

3．创建文档的标题，用<H3>…</H3>标签对：

```
<H3>Product Information</H3>
```

4．输入下面的表单部分，该部分包含了一个输入产品序列号的文本域、一个购买日期的文本域、一个购买方式的单选按钮组，还有一个复位按钮。

```
<FORM NAME="productInfo">
<B>Serial Number</B><BR>
<INPUT TYPE="text" NAME="serial" SIZE=50><P>
<B>Date Purchased</B><BR>
<INPUT TYPE="text" NAME="date" SIZE=20><P>
```

```
<B>Where did you purchase this product?</B><BR>
<INPUT TYPE="radio" NAME="where" VALUE="retail">Retail
        Store<BR>
<INPUT TYPE="radio" NAME="where" VALUE="catalog_mail">
Catalog/Mail Order<BR>
<INPUT TYPE="radio" NAME="where" VALUE="internet">
Internet Transaction<BR>
<INPUT TYPE="radio" NAME="where" VALUE="other">Other<P>
<INPUT TYPE="reset">
```

5．输入结束标签。

```
</FORM>
</BODY>
</HTML>
```

6．保存文件到 Tutorial.06 文件夹，文件名称为 ProductInfo.html。关闭该文件。在 Web 浏览器中打开该文件，然后单击“ Next ”按钮。文档底部的帧将打开 ProductInfo.html 文件，输出结果如图 6-19 所示。

7．关闭 Web 浏览器窗口。

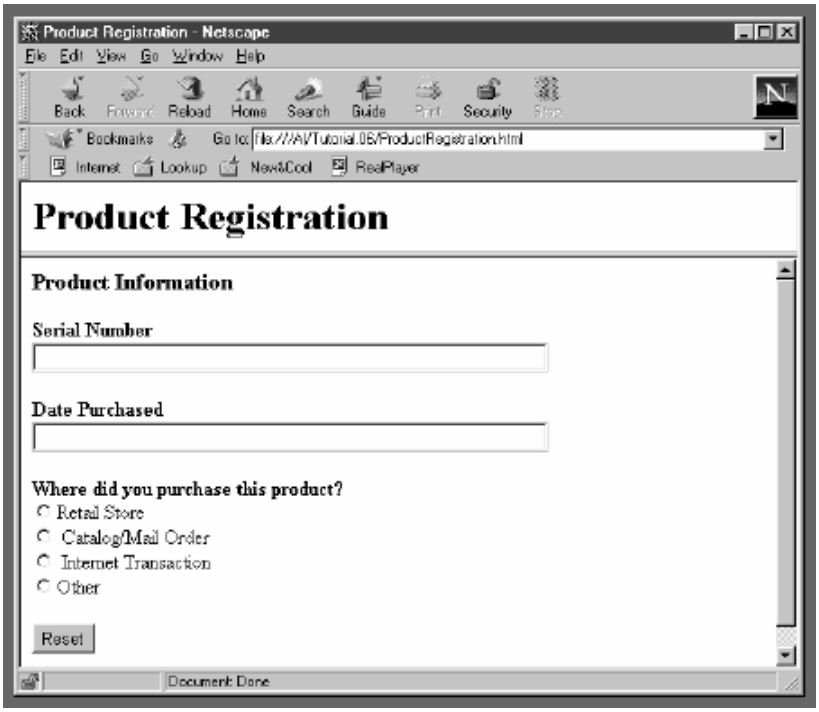


图 6-19　在 ProductRegistration.html 文件底部帧中的 ProductInfo.html 文件

提交按钮

TYPE 属性为 submit 的<INPUT>标签（<INPUT TYPE="submit">）创建一个提交按钮，该按钮将表单提交给服务器上的 CGI 脚本。创建表单的<FORM>标签的 ACTION 属性，指明了该表单将提交到哪个 URL 地址。在提交按钮中，还可以包括 NAME 和 VALUE 属性。如果没有包含 VALUE 属性，则提交按钮的标签会显示默认的 Submit Query 值。

下面的代码创建了一个有提交按钮的 Web 页面。

```
<H1>Video of the Month Club</H1>
<H3>Select the types of movies you like to see and
click the videocassette image.<BR>
A new movie will be sent to you every month.<H3>
<FORM ACTION="http://exampleurl/cgi-bin/cgi_program"
      METHOD="post" NAME="exampleForm">
<INPUT TYPE="checkbox" NAME="genre" VALUE="action">
Action
<INPUT TYPE="checkbox" NAME="genre" VALUE="adventure">
Adventure<BR>
<INPUT TYPE="checkbox" NAME="genre" VALUE="comedy">
Comedy
<INPUT TYPE="checkbox" NAME="genre" VALUE="drama">
Drama<BR>
<INPUT TYPE="checkbox" NAME="genre" VALUE="sci_fi">
Science Fiction
<INPUT TYPE="checkbox" NAME="genre" VALUE="western">
Westerns<P>
<INPUT TYPE="submit" NAME="submit_button" VALUE="Submit Query">
</FORM>
```

提示：用<INPUT TYPE="submit">创建的按钮的宽度取决于按钮的 VALUE 属性中字符的数目。

下面将往 ProductInfo.html 文件中添加一个提交按钮：

1．在文本或 HTML 编辑器中打开 ProductInfo.html 文件。

2. 在复位按钮<INPUT>标签后添加<INPUT TYPE="submit" VALUE="Submit Query">。

3. 保存并关闭文件 ProductInfo.html，然后在浏览器中打开文件 ProductRegistration.html。单击底部帧中的“Next”按钮，在其中打开 ProductInfo.html 文件。文件中的提交按钮应该和图 6-20 中的显示一样。实际上，如果单击提交按钮，不会发生任何事情，因为 ProductInfo.html 文件中的<FORM>标签没有指定 ACTION 属性。

4．关闭 Web 浏览器窗口。

图 6-20　添加了提交按钮的产品注册程序

图像提交按钮

TYPE 属性为 image 的<INPUT>按钮（<INPUT TYPE="image">）创建一个带有图像的按钮，该按钮往服务器的 CGI 脚本提交表单数据。图像提交按钮与前面的提交按钮执行同样的功能。用 SRC 属性来指定一个在按钮上显示的图像文件。在图像提交按钮中，可以包含 NAME、VALUE、MAXLENGTH 和 SIZE 属性。下面的代码创建了一个有图像提交按钮的 Web 页面，如图 6-21 所示。

```
<H1>Video of the Month Club</H1>
<H3>Select the types of movies you like to see and
click the videocassette image.<BR>
A new movie will be sent to you every month.<H3>
<FORM ACTION="http://exampleurl/cgi-bin/cgi_program"
      METHOD="post" NAME="exampleForm">
<INPUT TYPE="checkbox" NAME="genre" VALUE="action">
Action
<INPUT TYPE="checkbox" NAME="genre" VALUE="adventure">
Adventure<BR>
<INPUT TYPE="checkbox" NAME="genre" VALUE="comedy">
Comedy
<INPUT TYPE="checkbox" NAME="genre" VALUE="drama">
Drama<BR>
```

```
<INPUT TYPE="checkbox" NAME="genre" VALUE="sci_fi">
Science Fiction
<INPUT TYPE="checkbox" NAME="genre" VALUE="western">
Westerns<P>
<INPUT TYPE="image" SRC="videocas.jpg">
</FORM>
```

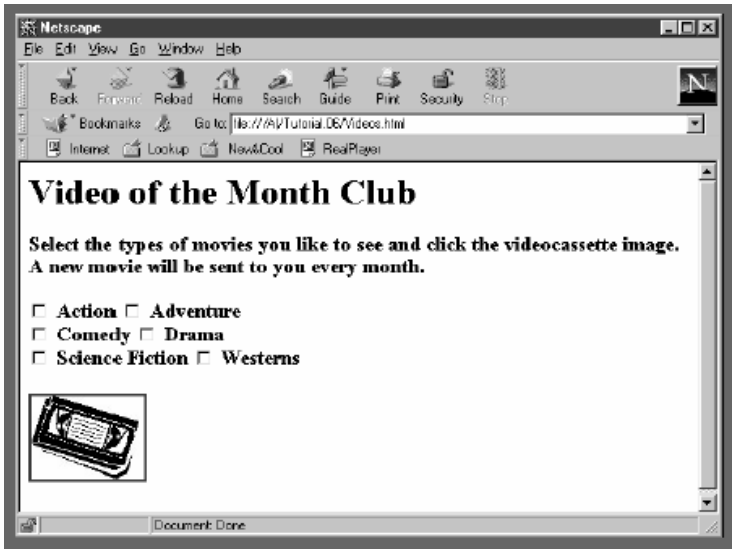


图 6-21　一个有图像提交按钮的表单的输出

提示：图像提交按钮也可以被用来创建服务器端的图像映射，此处的图像映射和在第 2 章里创建的客户端的南美图像映射相同。两者之间的区别在于服务器端的图像映射大部分是在服务器端完成的，而客户端的图像映射多数是在 Web 浏览器中完成的。

6.1.6　选择列表

标签对<SELECT>…</SELECT>用来创建一个选择列表，给用户提供固定的取值列表来进行选择。选择列表以下拉菜单的形式显示，同时可以带有滚动条（取决于列表中的条目的多少）。表 6-3 显示了<SELECT>标签的属性。

表 6-3　<SELECT>标签的属性

属　　性	描　　述
MULTIPLE	设置用户是否可以选择多于一个的项目
NAME	为选择列表指定一个名称
SIZE	决定一个选择列表按多少行显示。如果没有包括该属性或设为 1，那么选择列表按照下拉列表的形式显示

<OPTION>标签放置在<SELECT>…</SELECT>标签对之间，用来指定列表中的条目。表 6-4 列出了<OPTION>标签的属性。

表 6-4 <OPTION>标签的属性

属 性	描 述
SELECTED	可选属性，决定一个项目是否在列表载入时被初始选中
VALUE	提交到 CGI 脚本的值

下面的代码创建了两个选择列表。

```
<FORM ACTION="http://exampleurl/cgi-bin/cgi_program"
      METHOD="post" NAME="exampleForm">
This selection list displays a drop-down-style menu:<BR>
<SELECT NAME="music">
<OPTION VALUE="jazz">Jazz
<OPTION VALUE="classical">Classical
<OPTION VALUE="country">Country
<OPTION VALUE="rock" SELECTED>Rock
<OPTION VALUE="r&b">Rhythm and Blues
</SELECT><P>
This selection list displays 3 items:<BR>
<SELECT NAME="music" SIZE=3>
<OPTION VALUE="jazz">Jazz
<OPTION VALUE="classical" SELECTED>Classical
<OPTION VALUE="country">Country
<OPTION VALUE="rock">Rock
<OPTION VALUE="r&b">Rhythm and Blues
</SELECT>
</FORM>
```

下面将往 CustomerInfo.html 文件中添加一个选择列表，选择产品应用在何处，是学校、家庭还是家庭办公室中。

往 CustomerInfo.html 文件中添加一个选择列表：

1．在文本或 HTML 编辑器中打开 CustomerInfo.html 文件。

2．在复位按钮<INPUT>标签前面，添加下面的标签，创建选择产品应用场所的选择列表。

```
Where will you use this product?
<SELECT NAME="location">
      <OPTION VALUE="work">Work
      <OPTION VALUE="school">School
```

```
        <OPTION VALUE="home">Home
        <OPTION VALUE="home_office">Home Office
</SELECT><P>
```

3. 保存并关闭文件 CustomerInfo.html，然后在浏览器中打开文件 ProductRegistration.html。浏览器中的选择列表将如图 6-22 所示。

4．关闭 Web 浏览器窗口。

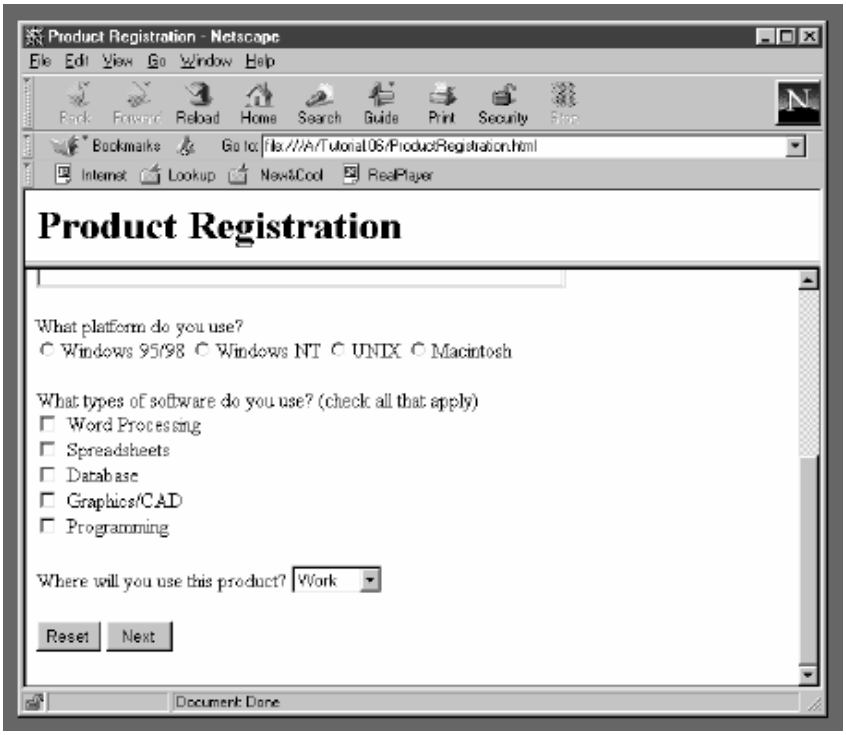


图 6-22　添加了选择列表的产品注册程序

6.1.7　多行文本输入域

<TEXTAREA>标签用来创建一个用户可以输入多行信息的文本域。由标签对<TEXTAREA>…</TEXTAREA>创建的是多行文本输入域。多行文本输入域有三个属性，列在表 6-5 中。

表 6-5　<TEXTAREA>标签的属性

属　　性	描　　述
NAME	为文本域指定一个名称
COLS	指定在文本域中显示的列数
ROWS	指定在文本域中显示的行数

在<TEXTAREA>…</TEXTAREA>标签对之间，可以放置的惟一项目是在表单载入时，想在文本域中显示的默认文本和字符。任何包括在<TEXTAREA>…</TEXTAREA>标签对中的字符，包括制表符和分段符都将包括在文本域中。例如，一行包含两个制表符的包含在<TEXTAREA>…</TEXTAREA>标签对之间的文本，显示在文本框中的文本也会按照两个制表符缩进。

下面的代码创建了一个包含 50 列 10 行的多行文本输入域，域中默认的文本是 Enter additional information here。

```
<FORM ACTION="http://exampleurl/cgi-bin/cgi_program"
      METHOD="post" NAME="exampleForm">
Comments<BR>
<TEXTAREA COLS=50 ROWS=10>
Enter additional information here
</TEXTAREA>
</FORM>
```

下面将添加一个多行文本输入域到 CustomerInfo.html 文件中。在该区域里，用户可以输入额外的注释。

添加一个多行文本输入域到 CustomerInfo.html 文件中：

1．在文本或 HTML 编辑器中打开 CustomerInfo.html 文件。

2．在复位按钮<INPUT>标签的前面，按回车，开始一个新的行。然后添加下面的代码，创建多行文本输入域。

```
Comments<BR>
<TEXTAREA NAME="comments" COLS=40 ROWS=5>
Enter any additional comments here
</TEXTAREA><P>
```

3. 保存并关闭文件 CustomerInfo.html，然后在浏览器中打开文件 ProductRegistration.html。结果应当如图 6-23 所示。

4．关闭 Web 浏览器的窗口。

6.1.8 总结

- ✧ 通用网关接口或者 CGI 是一个在 Web 服务器端的应用和 Web 页面间进行通信的简单协议。
- ✧ 环境变量是操作系统的一部分，与 JavaScript 的变量不同，仅仅是一个函数或程序的一部分。
- ✧ 一个处理 CGI 环境变量的 Web 服务器端的程序称为 CGI 脚本（或 CGI 程序），

并且被设计用来执行许多不同的功能。

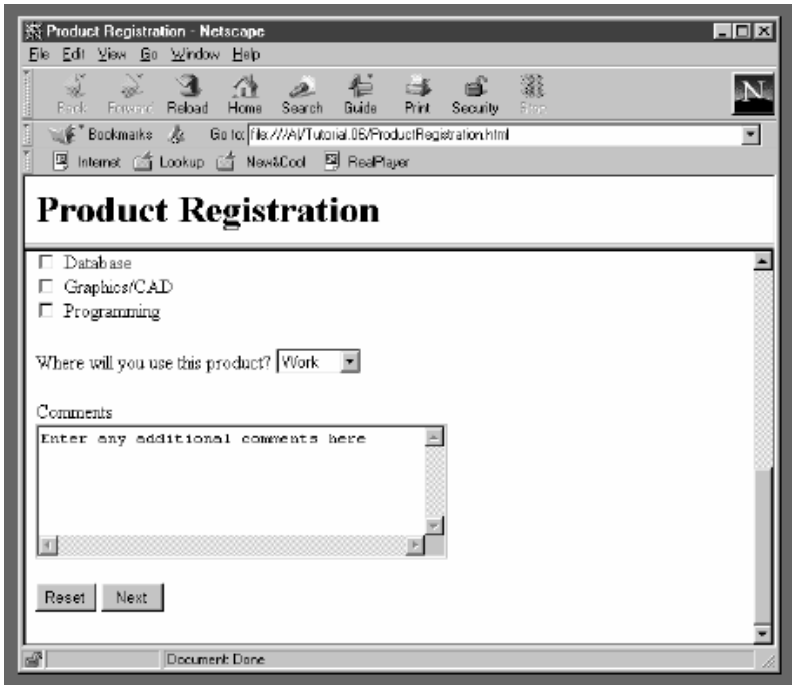


图 6-23　添加了一个多行文本输入域的产品注册程序

- ✧ CGI 脚本通常放置在 Web 服务器的一个 bin，或者 cgi-bin 目录中，以.cgi 为文件后缀。
- ✧ 标签对<FORM>…</FORM>用来设计一个表单，该标签对之间的所有文本和标签形成了一个表单。
- ✧ 标签对<FORM>…</FORM>之间，可以使用<INPUT>、<SELECT>和<TEXTAREA>三种标签。
- ✧ 任何一个用户可以输入数据（例如文本框）进行选择或更新的表单元素（例如单选按钮）称为域。
- ✧ 当往一个 CGI 脚本提交一个表单时，表单的数据按照 name=value 标签对的格式传输，name 和 value 取值于 NAME 和 VALUE 属性。
- ✧ <INPUT>标签用来创建收集信息的输入域，用不同类型的用户交互元素进行信息的收集。
- ✧ TYPE 属性为 text 的<INPUT>标签创建一个接收单行文本的简单的文本框。
- ✧ TYPE 属性为 password 的<INPUT>标签创建一个密码输入域，在密码输入域中输入的每个字符都是以星号的形式显示，以免有人从用户的后面看到用户输入的密码。
- ✧ TYPE 属性为 checkbox（<INPUT TYPE="checkbox">）创建一个可以被设置为 yes（选中）或者 no（未选中）的复选按钮。

- ✧ TYPE 属性为 reset（<INPUT TYPE="reset">）的复位按钮将清空表单的所有项目，并且将每个表单的元素设置为其由 VALUE 属性指定的初始值。
- ✧ TYPE 属性为 button 的<INPUT>标签（<INPUT TYPE="button">）创建一个与您在对话框中看到的“ OK ”和“ Cancel ”按钮相似的命令按钮。
- ✧ TYPE 属性为 submit 的<INPUT>标签（<INPUT TYPE="submit">）创建一个提交按钮，该按钮将表单提交给服务器上的 CGI 脚本。
- ✧ TYPE 属性为 image 的<INPUT>按钮（<INPUT TYPE="image">）创建一个带有图像的按钮，该按钮往服务器的 CGI 脚本提交表单数据。
- ✧ 标签对<SELECT>…</SELECT>用来创建一个选择列表，给用户提供固定的取值列表来进行选择。
- ✧ <TEXTAREA>标签用来创建一个用户可以输入多行信息的文本域。由标签对<TEXTAREA>…</TEXTAREA>创建的是多行文本输入域。

6.1.9 问题

1．下面哪个项目不是用表单创建的？

a. 客户登记

b. 调查表格

c. 在线定单系统

d. 以上所有的项目

2. CGI 是_____。

a. 一种类似于 VC 的高级编程语言

b. 一种类似于 JavaScript 的低级编程语言

c. 一种允许 Web 页面和 Web 服务器端程序通信的简单协议

d. 一种机器语言

3. CGI 脚本通常被放置在____。

a. Web 服务器上的 bin 或 cgi-bin 目录

b. 在本地操作系统的一个有用文件夹

c. 在 Navigator 或者 IE 的安装目录下

d. 在<SCRIPT>…</SCRIPT>标签对内

4. 在一个 HTML 文档中，可以创建多少个表单？

a. 1

b. 2

c. 任意多

d. 没有。在 HTML 文档中并不创建表单

5. <FORM>标签的 ACTION 属性_____。

a. 指定一个要执行的函数

b. 创建一个启动一个程序的按钮
c. 关闭 Web 浏览器窗口
d. 指定一个可以向其提交表单数据的 URL 地址
6. <FORM>标签的 METHOD 属性的默认提交方法是：
a. GET
b. POST
c. SEND
d. SUBMIT
7. <FORM>标签的 ENCTYPE 属性的默认数据格式是：
a. application/jpeg/gif
b. cgi.bin
c. application/x-www-form-urlencoded
d. text/plain
8. 一个表单的数据是如何提交到一个 CGI 程序的？
a. 用 value，name 标签对
b. 用 name=value 标签对
c. 用逗号分开的 value
d. 用分段符分隔的 value
9. 文本<INPUT>标签_____。
a. 显示一个静态标签
b. 创建一个简单的单行文本输入框
c. 创建一个接受多行文本的文本输入框
d. 是一种滚动的标题，用来在 Web 浏览器中显示消息
10. SIZE 属性是用在_____<INPUT>标签中。
a. button
b. image
c. text
d. submit
11. 在密码文本输入域输入的每个字符都显示为：
a. &
b. #
c. %
d. *
12. 在一个单选按钮组中，哪个属性指定其中一个按钮为默认值？
a. CHECKED
b. CHECK
c. SELECTED

d. DEFAULT

13. 下面关于复选按钮的的哪个描述是正确的？

a. 在一个组里，每次只能选择一个复选按钮

b. 可以选择需要的任意多的按钮

c. 当选择一个复选按钮时，在同组中的其他按钮同样会被选择

d. 复选按钮并不需要用户选择

14. 复位按钮的目的是：

a. 重新载入当前的 Web 页面

b. 将一个表单元素重新设置为默认值

c. 将所有的表单元素设置为默认值

d. 关闭并重启 Web 浏览器

15. 哪个类型的<INPUT>标签可以创建一个与在对话框中看到的“ OK ”和“ Cancel ”按钮相似的命令按钮？

a. radio

b. ok_cancel

c. dialog

d. button

16. 默认的提交按钮的标签是：

a. Submit

b. Query

c. Submit Query

d. Execute

17. 下面的哪个<INPUT>标签可以用来往一个 CGI 程序提交一个表单的数据？

a. image

b. radio

c. checkbox

d. button

18. 选择列表的内容是由哪个 HTML 标签决定的？

a. <SELECT>

b. <CONTENTS>

c. <ITEMS>

d. <OPTION>

19. 下面哪个是创建文本域的正确语法？

a. <TEXT COLS=50 ROWS=10></TEXT>

b. <TEXTAREA COLS=50 ROWS=10></TEXTAREA>

c. <TEXT SIZE=50></TEXT>

d. <TEXTAREA SIZE=50></TEXTAREA>

6.1.10　练习

1. 创建一个表单，用做软件开发调试报告。包括诸如产品名称、版本、硬件类型、操作系统、故障频率和推荐解决方案。保存文件为 BugReport.html 在文件夹 Tutorial.06 中。

2. 创建一个表单，用来跟踪、存档、管理一个专业职位候选人的会面情况。包括候选人的姓名、沟通能力、专业表现、计算机熟练程度、商务知识和会见者的注释。保存文件为 Interview.html 在文件夹 Tutorial.06 中。

3. 创建一个表单，管理商务发展联系。包括公司、位置和上次联系时间等域。保存文件为 BusinessDevelopment.html 在文件夹 Tutorial.06 中。

4. 创建一个学校旅行的同意表格。表单中包括诸如孩子的姓名、父母或监护人的签字、姓名、地址和孩子的医生的电话号码。保存文件为 ConsentForm.html 在文件夹 Tutorial.06 中。

5. 用表单创建您自己的在线信纸。用文本域作为收信者的地址，用一个文本域作为信的正体。在一个选择列表中包含常用的姓名。当用户单击列表中的一个姓名，用选择的人的 mail 信息填充收信人的几个输入域。用标准的 HTML 标签格式化信纸和您的个人信息。保存文件为 Stationery.html 在文件夹 Tutorial.06 中。

6.2　校验用户在表单的输入

本节目标

在本节里将学习：

- ✧　关于隐藏域的知识
- ✧　关于表单对象的知识
- ✧　如何引用表单和表单元素
- ✧　关于表单事件处理程序、方法和属性
- ✧　如何用 E-mail 发送表单的数据

6.2.1　表单隐藏域

表单隐藏域是一种特殊类型的表单元素，允许向用户隐藏信息。表单隐藏域用<INPUT>标签创建。一个 Web 浏览器不能显示表单隐藏域，当然用户也不能在浏览器窗口中进行编辑。通常，隐藏域用来暂存需要和余下的表单数据一起提交到服务器的信息，这些信息用户没有必要看到。存储在隐藏域中的信息的例子，包括计算的结果和其他将来编程用到的一些其他信息。创建隐藏域的语法和创建其他域的语法是相同的，都是用<INPUT>标签：<INPUT TYPE="hidden">。NAME 和 VALUE 是可以包括在表单隐藏域中的仅有属性。

图 6-24 包括了在第 3 章中创建的计算器程序的表单部分。程序被改造为具有可以保存

```
<HTML>
<HEAD>
<TITLE>Calculator</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
var inputString = "";
var count = 0;
function updateString(value) {
        inputString += value;
        document.Calculator.Input.value = inputString;
}
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
<BODY>
<CENTER>
<FORM NAME="Calculator">
<INPUT TYPE="text" NAME="Input" Size="22"><BR>
<INPUT TYPE="button" NAME="plus" VALUE=" + "
        onClick="updateString(' + ')">
<INPUT TYPE="button" NAME="minus" VALUE=" - "
        onClick="updateString(' - ')">
<INPUT TYPE="button" NAME="times" VALUE=" x "
        onClick="updateString(' * ')">
<INPUT TYPE="button" NAME="div" VALUE=" / "
        onClick="updateString(' / ')">
<INPUT TYPE="button" NAME="mod" VALUE=" MOD "
        onClick="updateString(' % ')"><BR>
<INPUT TYPE="button" NAME="zero" VALUE=" 0 "
       onClick="updateString('0')">
<INPUT TYPE="button" NAME="one" VALUE=" 1 "
        onClick="updateString('1')">
<INPUT TYPE="button" NAME="two" VALUE=" 2 "
        onClick="updateString('2')">
```

图 6-24　有存储功能的 Calculator.html

```
<INPUT TYPE="button" NAME="three" VALUE=" 3 "
        onClick="updateString('3')">
<INPUT TYPE="button" NAME="four" VALUE=" 4 "
        onClick="updateString('4')">
<INPUT TYPE="button" NAME="five" VALUE=" 5 "
        onClick="updateString('5')">
<INPUT TYPE="button" NAME="six" VALUE=" 6 "
        onClick="updateString('6')">
<INPUT TYPE="button" NAME="seven" VALUE=" 7 "
        onClick="updateString('7')">
<INPUT TYPE="button" NAME="eight" VALUE=" 8 "
        onClick="updateString('8')">
<INPUT TYPE="button" NAME="nine" VALUE=" 9 "
        onClick="updateString('9')"><BR>
<INPUT TYPE="button" NAME="point" VALUE=" . "
        onClick="updateString('.')">
<INPUT TYPE="button" NAME="clear" VALUE=" Clear "
        onClick="Input.value=''; inputString=''">
<INPUT TYPE="button" NAME="Calc" VALUE=" = "
        onClick="Input.value=eval(inputString)"><BR>
<INPUT TYPE="button" NAME="mem" VALUE=" M+ "
        onClick="storedValue.value=Input.value";>
<INPUT TYPE="button" NAME="recall" VALUE=" MRC "
       onClick="Input.value=storedValue.value";>
<INPUT TYPE="hidden" NAME="storedValue">
</FORM>
</CENTER>
</BODY>
</HTML>
```

续图 6-24　有存储功能的 Calculator.html

和引入数字的功能。图 6-25 显示了改动后的计算器。用来添加存储功能的三个新的表单元素的代码如下所示：

```
<INPUT TYPE="button" NAME="mem" VALUE=" M+    "
        onClick="storedValue.value=Input.value";>
<INPUT TYPE="button" NAME="recall" VALUE=" MRC "
        onClick="Input.value=storedValue.value";>
<INPUT TYPE="hidden" NAME="storedValue">
```

第一个新的按钮，命名为 mem，保存输入文本的值到名为 storedValue 的表单隐藏域中。第二个新按钮，命名为 recall，引入保存在 storedValue 隐藏域中的信息，并放置在输入文本框中。

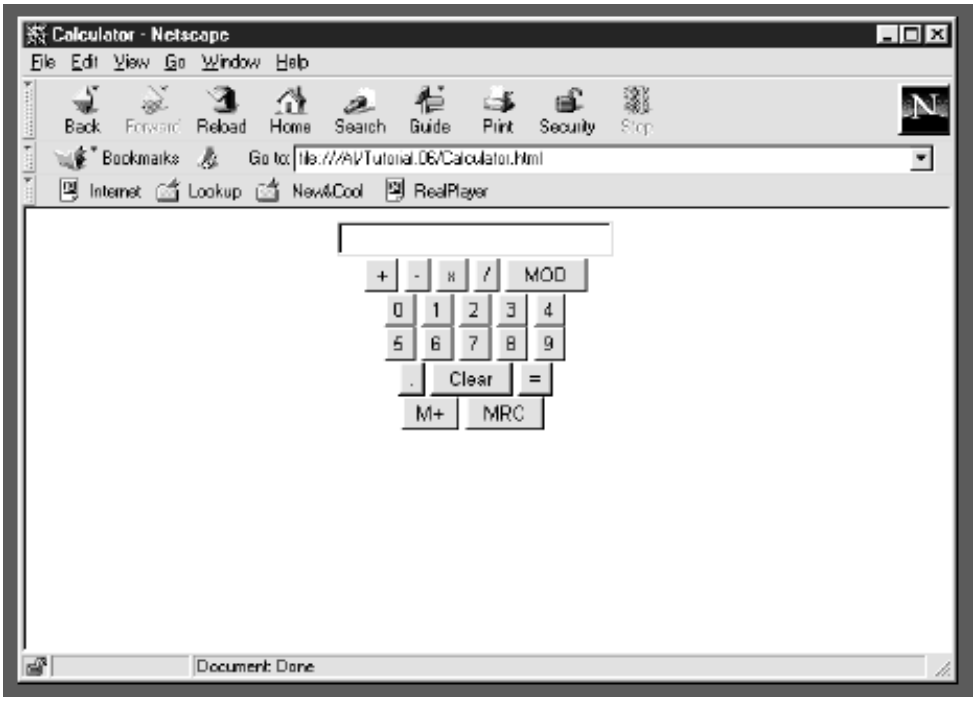


图 6-25　在 Web 浏览器中的 Calculator.html

下面，将添加表单隐藏域到 ProductRegistration.html 文档中，该文档是在第 1 章中创建的。ProductRegistration.html 文档有一个严重的问题：当单击“ Next ”按钮后，从客户信息表单转移到产品信息表单后，客户信息表单中的已经输入的信息就消失掉了。为了解决这个问题，我们准备往 TopFrame.html 文件中添加表单隐藏域。在客户信息表单和产品信息表单中的域将被拷贝并且保存在这些表单隐藏域中。首先，将添加隐藏域到 TopFrame.html 文件中。

添加隐藏域到 TopFrame.html 文件中：

1．用文本或 HTML 编辑器打开 TopFrame.html 文件。

2．在</BODY>标签前添加下面的表单和表单元素代码。每个表单元素的名称与 CustomerInfo.html 和 ProductInfo.html 中的表单元素相对应。

```
<FORM NAME="hiddenElements">
<INPUT TYPE="hidden" NAME="name">
<INPUT TYPE="hidden" NAME="address">
```

```
<INPUT TYPE="hidden" NAME="city">
<INPUT TYPE="hidden" NAME="state">
<INPUT TYPE="hidden" NAME="zip">
<INPUT TYPE="hidden" NAME="email">
<INPUT TYPE="hidden" NAME="password">
<INPUT TYPE="hidden" NAME="platform">
<INPUT TYPE="hidden" NAME="wp">
<INPUT TYPE="hidden" NAME="ss">
<INPUT TYPE="hidden" NAME="db">
<INPUT TYPE="hidden" NAME="gr">
<INPUT TYPE="hidden" NAME="pr">
<INPUT TYPE="hidden" NAME="comments">
<INPUT TYPE="hidden" NAME="location">
<INPUT TYPE="hidden" NAME="serial">
<INPUT TYPE="hidden" NAME="date">
</FORM>
```

3．保存并关闭 TopFrame.html 文件。

在学习如何编写代码将 CustomerInfo.html 和 ProductInfo.html 中的表单元素拷贝到隐藏域之前，首先必须学习如何用表单对象工作。

6.2.2　表单对象

JavaScript 经常用来对表单数据进行校验或处理，校验和处理通常在数据被提交到 CGI 脚本或服务器前进行。例如，客户可能用您站点上的在线定单表单订购商品。当客户单击提交按钮后，必须保证诸如发货地址、信用卡信息输入正确。虽然无法用 JavaScript 对某些类型的信息进行校验，例如信用卡号，但是可以用 JavaScript 来校验那些必须输入的域是否输入了数据，还是保留空白没有输入。要想用 JavaScript 来校验数据，必须使用表单对象的属性、事件和方法。

提示：如果一个表单需要复杂和高级的校验或处理，最好由服务器端的 CGI 脚本来完成。因为服务器通常比最终用户的桌面计算机或工作站要高级的多。

引用表单和表单元素

在第 5 章中介绍了 JavaScript 对象模型。回忆一下，模型中的一些对象包括另一些对象的数组。举例来说，窗口对象就包括一个 frames[]数组，该数组包含了在窗口中的所有帧。同样，文档对象也包括一个 forms[]数组，该数组包含了一个 HTML 文档中的所有表单。如果一个窗口没有包含任何表单，则 forms[]数组为空。文档中的第一个表单为 document.forms[0]，文档中的第二个表单为 document.forms[1]，依此类推。

类似地，表单对象也包含了一个 elements[]数组。可以用该数组引用表单中的每一个元素。表单中的元素按照它们在 JavaScript 解释器中遇到的顺序，分配到 elements[]数组中。要引用表单中的一个元素，就可以使用元素对应的 elements[]数组元素。举例来说，如果需要引用 HTML 文档表单中的第一个元素，用语句 document.forms[0].elements[0];即可。下面的代码显示了表单上的每个元素是如何安排到 elements[]数组中的。

```
<FORM NAME="exampleForm">
The following element is assigned to elements [0 ]
<INPUT TYPE="text" NAME="field1">
The following element is assigned to elements [1 ]
<INPUT TYPE="text" NAME="field2">
The following element is assigned to elements [2 ]
<INPUT TYPE="text" NAME="field3">
The following element is assigned to elements [3 ]
<INPUT TYPE="text" NAME="field4">
The following element is assigned to elements [4 ]
<INPUT TYPE="text" NAME="field5">
</FORM>
```

当用<FORM>标签创建一个表单时，可以用 NAME 属性为表单赋一个名称。可以用文档对象加上表单名称来引用一个表单。例如，对于一个 NAME 属性为 orderForm 的表单，可以用 document.orderForm 来引用。

同样，可以用 NAME 属性给每个表单元素一个名称。为什么给表单元素命名如此重要？因为，如果要将一个表单提交给 CGI 脚本，就需要给每个表单元素指定一个名称。同时，使用元素名称也是除了用 elements[]数组外的另一个引用元素的方法。例如，有一个文本域名为 textField1 在表单 orderForm 中，则可以用语句 document.orderForm.textField1;引用。

提示：在一个要提交到 CGI 脚本的表单中，可以不为表单指定 NAME 属性。<FORM>标签的 NAME 属性只是用来在 JavaScript 中更容易引用表单。

下面将往 CustomerInfo.html 和 ProductInfo.html 文件中添加将表单域的值拷贝到隐藏域中的代码。首先添加 CustomerInfo.html 文件。

往 CustomerInfo.html 文件中添加将表单域的值拷贝到隐藏域中的代码：

1．在文本或 HTML 编辑器中打开 CustomerInfo.html 文件。

2．在函数 nextForm()中，在代码 location.href="ProductInfo.html"之前，添加一行新的代码，代码如下所示，将 CustomerInfo.html 中的表单域的值拷贝到 TopFrame.html 中的隐藏域中。在代码中，用窗口对象的 parent 属性来引用每个帧的名字。代码看上去可能比较复杂，但是您已经明白了每行代码的构造。例如，在第一行代码中，parent 是指定义了帧集合的 Registration.html 文档，topframe 指在窗口顶部包含 TopFrame.html 文档的帧，document 指 TopFrame.html 文件中的文档对象。在语句中的下一个项目是 TopFrame.html

文件中的名为 hiddenElements 的表单，后面的 name 指定了在表单 hiddenElements 中的名称输入域。最后的部分是 value，返回在 name 输入域中的信息。等号右侧的语句和左侧的语句相同，除了不需要 parent 属性或帧的名称（原因是指定的是当前帧）。

```
parent.topframe.document.hiddenElements.name.value =
    document.customerInfo.name.value;
parent.topframe.document.hiddenElements.address.value =
    document.customerInfo.address.value;
parent.topframe.document.hiddenElements.city.value =
    document.customerInfo.city.value;
parent.topframe.document.hiddenElements.state.value =
    document.customerInfo.state.value;
parent.topframe.document.hiddenElements.zip.value =
    document.customerInfo.zip.value;
parent.topframe.document.hiddenElements.email.value =
    document.customerInfo.email.value;
parent.topframe.document.hiddenElements.password.value =
    document.customerInfo.password.value;
parent.topframe.document.hiddenElements.platform.value =
    document.customerInfo.platform.value;
if (document.customerInfo.wp.checked == true)
    parent.topframe.document.hiddenElements.wp.checked =
    true;
if (document.customerInfo.ss.checked == true)
    parent.topframe.document.hiddenElements.ss.checked =
    true;
if (document.customerInfo.db.checked == true)
    parent.topframe.document.hiddenElements.db.checked =
    true;
if (document.customerInfo.gr.checked == true)
    parent.topframe.document.hiddenElements.gr.checked =
    true;
if (document.customerInfo.pr.checked == true)
    parent.topframe.document.hiddenElements.pr.checked =
    true;
parent.topframe.document.hiddenElements.location.value =
    document.customerInfo.location.value;
parent.topframe.document.hiddenElements.comments.value =
    document.customerInfo.comments.value;
```

3．保存并关闭 CustomerInfo.html 文件。

4．在 Web 浏览器中打开文件 ProductRegistration.html，并测试程序。

5．关闭 Web 浏览器窗口。

下面将学习添加验证用户输入的代码，该代码在用户单击“ Next ”按钮时，对用户必须输入的域进行确认。必须输入的域包括姓名、地址、城市、州和密码。

添加验证用户输入的代码：

1．在文本或 HTML 编辑器中打开 CustomerInfo.html 文件。

2．在语句 location.href="ProductInfo.html"前面添加下面的语句。其中的 if 语句用逻辑或操作符||进行判断，以保证所有要求的输入域都已经输入。如果其中有一个域等于空串""，则弹出一个警告对话框，提示用户必须填写这些输入域。如果所有的域都被输入了，则执行 else 语句中的 location.href="ProductInfo.html";的语句。

```
if (parent.topframe.document.hiddenElements
.name.value == ""
|| parent.topframe.document.hiddenElements
        .address.value == ""
|| parent.topframe.document.hiddenElements
        .city.value == ""
|| parent.topframe.document.hiddenElements
        .state.value == ""
|| parent.topframe.document.hiddenElements
        .zip.value == ""
|| parent.topframe.document.hiddenElements
        .password.value == "")
        alert("You must fill in the name, address, city,
                state, zip, and password fields.");
else
```

3．保存并关闭 CustomerInfo.html 文件。

4．在 Web 浏览器中打开文件 ProductRegistration.html。留下若干要求的域不输入，然后单击“ Next ”按钮，测试是否能够弹出警告对话框。

5．单击“ OK ”按钮关闭警告对话框。

6．填写所有要求输入的域，检验单击“ Next ”按钮后是否能够正确地引入 ProductInfo.html。

7．关闭 Web 浏览器窗口。

还必须添加往 TopFrame.html 文件中的隐藏域拷贝 ProductInfo.html 表单域取值的代码，并且对需要输入的域不能为空进行确认。因为 ProductInfo.html 文件没有包含“ Next ”按钮，需要将代码添加到提交按钮的 onSubmit 事件处理器中。该操作将在后面进行。

表单事件处理器

在第 2 章中，学习了许多可以在 JavaScript 中使用的事件处理器。另外的两个事件处理器是 onSubmit 和 onReset，用在<FORM>标签中。在一个提交按钮或图像提交按钮往 CGI

脚本提交一个表单时，就执行 onSubmit 事件处理器。onSubmit 事件处理器经常用来校验和检验提交到服务器的表单数据的有效性。当一个表单中的复位按钮被单击后，onReset 事件处理器将被执行。可以用 onReset 事件处理器来确定用户是否真正想将一个表单的内容复位。onSubmit 和 onReset 事件都被放置在<FORM>标签的结束括号前面。下面的代码显示了一个有 onSubmit 和 onReset 事件处理器的表单。

```
<FORM ACTION="http://exampleurl/cgi-bin/cgi_program"
      METHOD="post" NAME="exampleForm"
      onSubmit="JavaScript statements ;"
      onReset="JavaScript statements ;">
```

当使用 onSubmit 和 onReset 事件处理器的时候，可以返回一个为真或假的值，取值的依据是该表单是否需要提交或复位。例如，下面的代码根据用户是否单击了确定对话框中的确定或取消按钮，返回一个为真或假的的值。如果用户单击“ OK ”按钮，确定对话框返回一个为真的值，并执行 onSubmit 事件。如果用户单击“ Cancel ”按钮，确定对话框返回一个为假的值，不执行 onSubmit 事件。

```
<FORM ACTION="http://exampleurl/cgi-bin/cgi_program"
      METHOD="post" NAME="exampleForm"
      onReset="return confirm(
      'Are you sure you want to reset the form?')">
```

图 6-26 显示了一个包括 onSubmit 和 onReset 事件处理器的程序。当 onSubmit 事件处理器执行时，程序用一个函数校验表单的数据，并且使用另一个函数来确定用户是否真正想要复位一个表单。

下面将往 CustomerInfo.html 和 ProductInfo.html 文件中添加 onReset 事件处理器，来确定用户是否真正想要复位两个文件中的表单。

往 CustomerInfo.html 和 ProductInfo.html 文件中添加 onReset 事件处理器：

1. 在文本或 HTML 编辑器中打开 CustomerInfo.html。

2. 在文件的<HEAD>部分，在<SCRIPT>…</SCRIPT>标签对中的函数 nextForm()前面添加下面的函数 confirmReset()：

```
function confirmReset() {
      var resetForm = confirm(
          "Are you sure you want to reset the form?");
      if (resetForm == true)
          return true;
      return false;
}
```

3. 然后在<FORM>标签的结束括号前，添加 onReset="return confirmReset();"。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
function confirmSubmit() {
        var sendForm = confirm(
                "Are you sure you want to submit the form?");
        if (sendForm == true)
                return true;
        return false;
}
function confirmReset() {
        var resetForm = confirm(
                "Are you sure you want to reset the form?");
        if (resetForm == true)
                return true;
return false;
}
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
<FORM ACTION="http://exampleurl/cgi-bin/cgi_program"
        METHOD="post" NAME="exampleForm"
        onSubmit="return confirmSubmit();"
        onReset="return confirmReset();">
Name<BR>
<INPUT TYPE="text" NAME="name" SIZE=50><BR>
Address<BR>
<INPUT TYPE="text" NAME="address" SIZE=50><BR>
City, State, Zip<BR>
<INPUT TYPE="text" NAME="city" SIZE=38>
<INPUT TYPE="text" NAME="state" SIZE=2 MAXLENGTH=2>
<INPUT TYPE="text" NAME="zip" SIZE=5 MAXLENGTH=5><BR>
<INPUT TYPE="reset">
<INPUT TYPE="submit">
</FORM>
```

图 6-26　包含 onSubmit 和 onReset 事件处理器的程序

4．保存并关闭文件 CustomerInfo.html。

5．在文本或 HTML 编辑器中打开 ProductInfo.html 文件，并且在<HEAD> 部分添加

<SCRIPT>…</SCRIPT>标签对。在<SCRIPT>…</SCRIPT>标签对中添加与文件 Customer-Info.html 中相同的 confirmReset()。

6. 在 <FORM> 标签中的结束括号前添加事件处理器代码 onReset="return confirmReset();"。

7. 保存并关闭文件 ProductInfo.html。

8. 在 Web 浏览器中打开文件 ProductRegistration.html。在客户信息表单中输入一些数据，然后单击复位按钮，看看是否能看到警告信息。单击“ OK ”按钮关闭警告对话框。在客户信息表中添入需要的信息，然后单击“ Next ”按钮。然后测试 ProductInfo.html 文件中的复位按钮。单击“ OK ”按钮关闭警告对话框。

9. 关闭 Web 浏览器窗口。

下面将往 ProductInfo.html 文件中添加 onSubmit 事件，用来校验文件中输入数据的有效性，并且将每个表单域中的数据拷贝到 TopFrame.html 文件中对应的隐藏表单域中。

往 ProductInfo.html 文件中添加 onSubmit 事件：

1. 在文本或 HTML 编辑器中打开 ProductInfo.html 文件。

2. 在<FORM>标签的结束括号前，添加下面的 onSubmit 事件代码 onSubmit="return submitForm();"。

3. 在<SCRIPT>…</SCRIPT>标签对中，在 confirmReset()前，添加 function submitForm(){代码段，作为函数 submitForm()的开始。

4. 然后，往 submitForm()函数中添加下面的语句，将 ProductInfo.html 中的表单域拷贝到 TopFrame.html 中的对应隐藏表单域中。

```
parent.topframe.document.hiddenElements.serial.value=
        document.productInfo.serial.value;
parent.topframe.document.hiddenElements.date.value =
        document.productInfo.date.value;
```

5. 接着上面的语句，添加如下的代码，确认用户是否已经填写了要求的信息。如果用户已经填写了要求的表单域，返回一个为真的值，并且提交表单数据。如果返回为假，则取消 onSubmit 事件的执行。

```
if (parent.topframe.document.hiddenElements.serial.
        value == ""
|| parent.topframe.document.hiddenElements.date.
        value == "") {
        alert(
"You must fill in the date and serial number.");
        return false;
}
```

```
<HTML>
<HEAD>
<TITLE>Counting Contest</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
var apples = 7; var runners = 6; var pushpins = 3;
var applesCorrect; var runnersCorrect; var pushpinsCorrect;
function confirmAnswers(picture, answer) {
        if (picture == "apples") {
                if (answer == "7") applesCorrect = true;
                else {
                        alert("Sorry! You must start over.");
                        document.contestForm.reset();
                }
        }
        if (picture == "runners") {
                if (answer == "6") runnersCorrect = true;
                else {
                        alert("Sorry! You must start over.");
                        document.contestForm.reset();
                }
   }
        if (picture == "pushpins") {
                if (answer == "3") pushpinsCorrect = true;
                else {
                        alert("Sorry! You must start over.");
                        document.contestForm.reset();
                }
}
if (applesCorrect == true && runnersCorrect == true &&
pushpinsCorrect == true) {
                alert("Congratulations! You will be entered in the
contest.");
                document.contestForm.submit();
        }
}
```

图 6-27　包含有 submit()和 reset()方法的程序

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
<BODY>
<H1>How Many Items Can You Count?</H1>
<H2>Count the number of items in each picture.
If you count all the items correctly, you will be entered
into a drawing to win a free trip.</H2>
<H3>Watch out! If you enter the wrong number of items for
any picture, you will need to start over from the
beginning.</H3>
<FORM ACTION="http://exampleurl/cgi-bin/cgi_program"
      METHOD="post" NAME="contestForm">
<IMG SRC="apples.jpg"><IMG SRC="runners.jpg"><IMG
SRC="pushpins.jpg"><P>
Apples: <INPUT TYPE="text" SIZE="15"
onChange="confirmAnswers('apples', this.value);">
Runners: <INPUT TYPE="text" SIZE="15"
onChange="confirmAnswers('runners', this.value);">
Pushpins: <INPUT TYPE="text" SIZE="15"
onChange="confirmAnswers('pushpins', this.value);">
</FORM>
</BODY>
</HTML>
```

续图 6-27　包含有 submit()和 reset()方法的程序

6. 添加 submitForm()函数的结束括号。

7. 保存 ProductInfo.html 文件。注意 TopFrame.html 文件中包含了收集表单数据的隐藏表单域。在测试 submitForm()函数前，需要添加提交 TopFrame.html 文件中的隐藏表单域的语句。如果需要用 JavaScript 提交表单内容，需要学习下一节中的表单方法。

表单方法

表单对象仅包含两个方法：submit()和 reset()。submit()方法用来提交一个表单数据，而不必使用提交<INPUT>标签。reset()方法用来复位一个表单，而不必使用复位<INPUT>标签。submit()和 reset()方法执行与提交和复位按钮同样的功能。但是，当使用 submit()和 reset()方法时，并不执行 onSubmit 和 onReset 事件。任何校验数据的代码必须包含在调用 submit()和 reset()方法的代码中。

submit()和 reset()方法用在没有提交和复位按钮的表单中。图 6-27 显示了用 submit()和 reset()方法的一个程序。该程序包含了一个游戏，在游戏里，用户必须正确地对若干项目组

进行计数。如果用户数错了一组的计数，则 reset()方法清除表单，用户必须重新开始。如果用户计数正确，表单就会被自动提交。程序用 onChange 事件处理器来调用 confirmAnswers()函数。该函数检查传入的 picture 和 answer 参数，然后采取合适的动作。在函数中的最后一个 if 语句，检查表示三个项目的三个全局变量，看它们是否已经被设置为真。如果三个全局变量已经被置为真，则提交表单。图 6-28 显示了该程序在 Web 浏览器窗口中的输出。

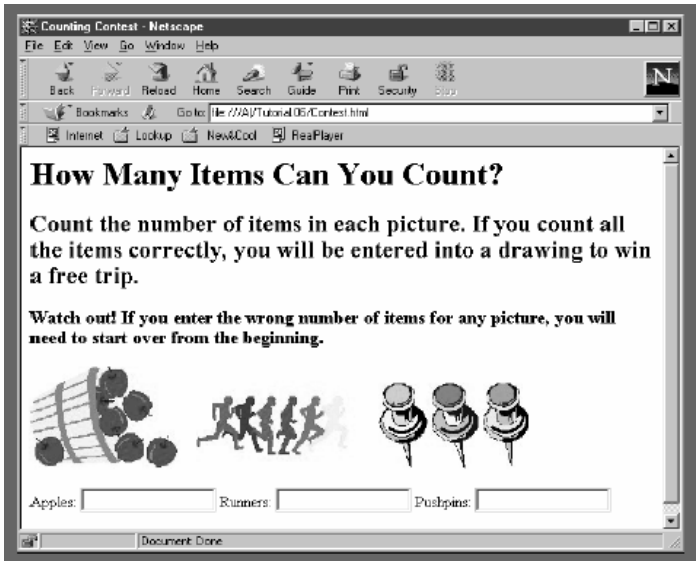


图 6-28　包含有 submit()和 reset()方法的程序输出

下面，将往 ProductInfo.html 文件中添加用 submit()方法提交 TopFrame.html 文件中表单内容的语句：

1．在文本或 HTML 编辑器中打开 ProductInfo.html 文件。

2．在 submitForm()函数的结束括号前，添加下面的 else 子句，该子句提交 TopFrame.html 文件中的隐藏表单域中的数据。注意向 onSubmit 事件处理器返回一个为假的值，以避免 ProductInfo.html 文件提交它自己的表单域。

```
else {
parent.topframe.document.hiddenElements.submit();
return false;
}
```

3．保存并关闭 ProductInfo.html 文件。

4．在 Web 浏览器中打开文件 ProductRegistration.html。在客户信息表单中输入要求的数据，然后单击“ Next ”按钮。在没有输入任何产品信息表单的数据情况下，单击“ Submit ”按钮。应该看到一个警告对话框。

5．单击“ OK ”按钮来关闭警告对话框，然后关闭 Web 浏览器窗口。

提示：如果输入数据并单击“ Submit ”按钮，并不会发生任何事情，因为我们并没有一个真正的 Web 服务器来使用。在本节的结束部分，将添加一段往一个 E-mail 地址提交数据的代码。

表单属性

对应于<FORM>标签的属性，表单对象包括七个属性。表单对象还有包含其表单元素的属性。表 6-6 列出了表单的属性。在第一节里，您一定对其中的大部分属性非常熟悉了。

表 6-6　表单对象的属性

属　　性	说　　明
action	表单的数据将被提交到的 URL
method	表单的数据用何种方式提交：GET 或 POST
enctype	数据提交的格式
target	显示从服务器返回的结果的窗口
name	表单的名称
elements[]	包含了表单元素的数组
length	一个表单中的元素数目

提示：表单对象的属性，除了 name、elements[]、length 属性，都可以用 JavaScript 代码修改。

长度（length）属性对检索一个表单中的元素数目很有用处。图 6-29 包括了一段程序，该程序用 length 属性来检测一个表单域是否已经有了输入。程序中的 onSubmit 事件调用了一个名为 confirmFields()的函数。该函数用一个 while 语句循环检测 elements[]数组。如果在 elements[]数组中的一个元素为空，则一个名为 missingFields 的变量将被加 1。当 while 语句执行结束后，用一个 if 语句检查 missingFileds 的值是否大于 0。如果大于 0，就弹出一个警告对话框，提示用户有多少个表单域没有输入。需要注意的是，在 length 属性里去掉了一个元素，因为提交按钮不用检查输入情况，所以去掉了提交按钮的计数。

6.2.3　用 E-mail 发送表单数据

目前为止，看到的表单都是将数据提交到服务器上的 CGI 脚本。另一个选择是将表单数据发送到一个 e-mail 地址。将表单数据发送到 E-mail 地址远比编写 CGI 脚本程序要简单的多。与依赖于一个复杂的 CGI 脚本来处理数据不同，需要一个 E-mail 消息的容器来处理数据。举例来说，一个包含有产品在线订单的站点中，用户单击了提交按钮提交表单数据后，订单的数据被传送到负责处理订单的职员的 E-mail 地址中。对每天处理成百上千的订单的大公司来说，将所有的订单都 E-mail 给一个单独的员工不是一个好的解决方法。但对于一个小公司或 Web 站点来说，通常没有太多的订单，E-mail 一个表单数据是一个很好的

解决方案。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
function confirmFields() {
        var count = 0;
        var missingFields = 0;
        while (count < document.exampleForm.length - 1) {
                if (document.exampleForm.elements [count ].value == "")
                        missingFields = ++missingFields;
                        ++count;
        }
        if (missingFields > 0) {
                alert("You are missing " + missingFields + " out of " +
                        (document.exampleForm.length - 1) + " fields.");
                return false;
        }
        return true;
}
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
<FORM ACTION="http://exampleurl/cgi-bin/cgi_program"
        METHOD="post" NAME="exampleForm"
        onSubmit="confirmFields();">
Name<BR>
<INPUT TYPE="text" NAME="name" SIZE=50><BR>
Address<BR>
<INPUT TYPE="text" NAME="address" SIZE=50><BR>
City, State, Zip<BR>
<INPUT TYPE="text" NAME="city" SIZE=38>
<INPUT TYPE="text" NAME="state" SIZE=2 MAXLENGTH=2>
<INPUT TYPE="text" NAME="zip" SIZE=5 MAXLENGTH=5><BR>
<INPUT TYPE="submit">
</FORM>
```

图 6-29　表单域输入校验程序

要提交表单数据到 E-mail 地址，要把<FORM>标签中的 ACTION 属性的 CGI 脚本的 URL 地址，替代为 E-mail 地址：mailto:email_address。图 6-30 显示了一个示例代码，代码创建一个表单，表单数据将被 E-mail 到一个虚拟的 E-mail 地址：john_howe@exampledomain.com。

```
<HTML>
<HEAD>
<TITLE>Customer Information</TITLE>
</HEAD>
<BODY>
<H2>Customer Information</H2>
<FORM ACTION="mailto:john_howe@exampledomain.com"
      METHOD="post" ENCTYPE="text/plain"
NAME="customer_information">
Name<BR>
<INPUT TYPE="text" NAME="name" SIZE=50><BR>
Address<BR>
<INPUT TYPE="text" NAME="address" SIZE=50><BR>
City, State, Zip<BR>
<INPUT TYPE="text" NAME="city" SIZE=38>
<INPUT TYPE="text" NAME="state" SIZE=2 MAXLENGTH=2>
<INPUT TYPE="text" NAME="zip" SIZE=5 MAXLENGTH=5><BR>
E-Mail<BR>
<INPUT TYPE="text" NAME="email" SIZE=50><P>
<INPUT TYPE="reset">
<INPUT TYPE="submit">
```

图 6-30　E-mail 表单数据的程序

当传送一个表单数据到一个 E-mail 地址时，应使用 ENCTYPE 属性为 text/plain。ENCTYPE 属性 text/plain 保证数据以可读的格式传送到 E-mail 地址。图 6-31 显示了一个 E-mail 消息，该消息是 john_howe@exampledomain.com 接收到的图 6-30 中的程序提交的数据信息。该 E -mail 消息被显示在 Outlook Express 中。

如果忽略了 ENCTYPE 属性 text/plain，则表单数据按照默认的 application/x-www-form–urlencoded 格式传送。如果按照默认格式传送，则数据阅读和处理起来将比较困难。图 6-32 显示了以 application/x-www-form–urlencoded 格式从同一个程序传送来的数据。

提示：可以用在图 6-32 中显示的数据。但是，必须写一个专门的转换程序将其转换为可用的格式。

E-mail 表单数据的缺陷是并不是所有的 Web 浏览器都支持 mailto:email_address 选项。此外，mailto:email_address 选项的特性并不可靠。某些支持 E-mail 表单数据的 Web 浏览器有可能没法将数据正确地放置到 E-mail 消息中。如果写了一个 E-mail 表单数据的 Web 页面，一定要确保在使用前进行全面测试。

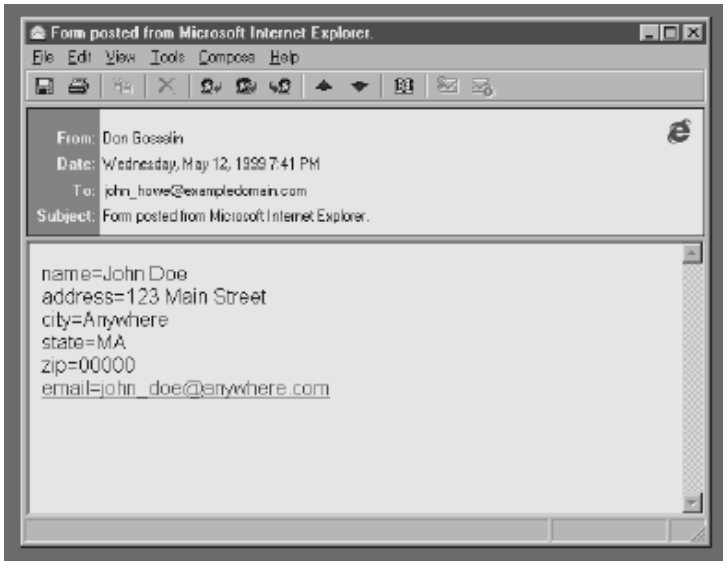


图 6-31 E-mail 表单数据在 Outlook Express 显示的消息

```
name=John+Doe&address=123+Main+Street&city=Anywhere&state=MA
&zip=00000&email=john_doe@anywhere.com
```

图 6-32 以 application/x-www-form–urlencoded 格式 E-mail 来的数据

提示：当用户单击一个将被 E-mail 的表单中的提交按钮，可能会收到一个安全性警告，或者要求是否编辑该 E-mail，取决于他们的 E-mail 应用是如何配置的。

下面将变更产品注册程序中的<FORM>标签，以便表单数据可以随时被提交到您的 E-mail 地址。为了保证该练习正确地执行，必须用 IE4.0 或 Navigator4.0 更高版本。

变更产品注册程序中的<FORM>标签，以便用 E-mail 来提交表单数据：

1．在文本或 HTML 编辑器中打开 TopFrame.html 文件。

2．在<FORM>标签的关闭括号前，添加下面的属性代码。用自己的 E-mail 地址取代其中的 email_address。

```
ACTION=mailto:email_address  METHOD="post"
ENCTYPE="text/plain"
```

3．保存并关闭 TopFrame.html 文件。在 Web 浏览器中打开文件 Product-Registration.html。填写每个表单中的域，然后提交表单。会收到一个警告对话框或者选择是否编辑 E-mail 消息。

4．稍微等几分钟，克服 Internet 传输的延迟，将会从您的 E-mail 地址收到新的消息，在收到 E-mail 后，检查其中的表单数据是否正确。

5．关闭 E-mail 程序和 Web 浏览器窗口。

6.2.4　总结

- ✧ 表单隐藏域用<INPUT>标签创建，并且允许向用户隐藏信息。
- ✧ 在 JavaScript 中，可以使用表单对象的属性、方法和事件来校验表单的信息。
- ✧ 文档对象包括一个 forms[]数组，该数组包含了一个 HTML 文档中的所有表单。
- ✧ 一个表单中的每个元素都可以用表单对象的 elements[]数组来引用。
- ✧ 在一个提交按钮或图像提交按钮往 CGI 脚本提交一个表单时，就执行 onSubmit 事件处理器。
- ✧ 当一个表单中的复位按钮被单击后，onReset 事件处理器将被执行。
- ✧ submit()方法用来提交一个表单数据，而不必使用提交<INPUT>标签。
- ✧ reset()方法用来复位一个表单，而不必使用复位<INPUT>标签。
- ✧ 对应于<FORM>标签的属性，表单对象包括几个属性。表单对象还有对应于其包含的表单元素的属性。
- ✧ 长度（length）属性对检索一个表单中的元素数目很有用处。
- ✧ 一个表单的数据可以被提交到一个 E-mail 地址，以代替将数据提交到服务器上的 CGI 脚本的方法。

6.2.5　问题

1. 下面哪个是创建一个表单隐藏域的正确代码？

a. <HIDDEN VALUE=" text to be hidden">

b. <HIDDEN>text to be hidden</HIDDEN>

c. <INPUT TYPE="hidden" NAME="storedValue">

d. <INPUT TYPE="hide" NAME="storedValue">

2. 在 JavaScript 中，可以使用____对象的属性、方法和事件来校验表单的信息。

a. Form

b. Document

c. Window

d. Data

3. 如果一个 HTML 文档中的第一个表单命名为 myForm，下面哪个 JavaScript 代码可以用来正确地引用该表单？

a. myForm.document

b. document.frames[1]

c. document.forms[0]

d. document.forms[1]

4. 表单中的每个元素都可以用_____数组来引用。

a. elements[]

b. input[]

c. components[]

d. forms[]

5. 对于<FORM>标签中的 NAME 属性，下面哪个语句是正确的？

a. 在<FORM>标签中不能包括 NAME 属性

b. 如果表单将被提交给一个 CGI 脚本，则<FORM>标签中的 NAME 属性是必须的

c. 如果在<FORM>标签中不包括 NAME 属性，则就没法在 JavaScript 中引用表单

d. <FORM>标签中的 NAME 属性只是为了在 JavaScript 中更容易地引用表单

6. 一个 HTML 文档包括两个帧。第一个帧命名为 leftFrame，第二个帧命名为 rightFrame。右边的帧包括一个名为 myForm 的表单，表单包括一个名为 myCheckbox 的复选按钮，该按钮是表单中的第二个元素。如果从 leftFrame 帧中将 myCheckbox 的值置为真，下面哪个语句是正确的？

a. parent.rightFrame.elements [0].checked =true;

b. parent.document.rightFrame.elements [1].checked =true;

c. myForm.myCheckbox.value =true;

d. myForm.myCheckbox.checked =true;

7. onSubmit 事件处理器常用来_____。

a. 在提交到一个 CGI 程序之前，将表单保存到一个本地文件

b. 定期地刷新 Web 浏览器的窗口，目的是显示一个表单的大多数当前数据

c. 当一个 Web 浏览器窗口打开或关闭时，检查新的 e-mail 消息

d. 在表单数据传送到服务器之前，对表单数据进行检查和有效性校验

8. 下面哪个是 onReset 事件处理器的正确代码？

a. <BUTTON onReset="functionName()";>

b. <RESET onReset="functionName()";>

c. <FORM onReset="functionName()";>

d. <SUBMIT onReset="functionName()";>

9. 为了取消 onSubmit 和 onReset 事件处理器的执行，需要执行哪个 JavaScript 代码？

a. 返回一个为假的值

b. 返回一个为真的值

c. 执行 stop 或 quit 方法

d. 什么也不做。只要 JavaScript 解释器遇到相应的事件处理器，onSubmit 和 onReset 事件就被取消

10. 表单对象包括两个方法：submit()方法和_____方法。

a. resubmit()

b. execute()

c. process()

d. reset()

11. _____属性返回表单元素的数目。

a. length

b. number

c. elements

d. fields

12. 把一个表单的数据传送到一个 E-mail 地址时，ENCTYPE 属性应该采取哪种取值？

a. application/jpeg/gif

b. cgi.bin

c. application/x-www-form-urlencoded

d. text/plain

6.2.6　练习

1．创建一个五金店的购买表单。包括标准货物的选择列表，例如铁锤、扳手和其他工具。用 JavaScript 代码来校验客户必须填写的诸如单据和运输信息。保存文件到 Tutorial.06 文件夹，文件名称为 PurchaseOrder.html。

2. 创建一个 6 年级的数学测验。用表单隐藏域来保存测验的答案。将提交按钮命名为 Score Quiz。当学生单击提交按钮时，执行 onSubmit 事件处理器，并检测他们是否回答了所有的问题。如果学生回答了所有的问题，用隐藏域中的答案来计算他们的分数。如果没有回答所有的问题，取消 onSubmit 事件的执行，并用警告对话框提示用户回答所有的问题。保存文件到 Tutorial.06 文件夹，文件名称为 ScoreQuiz.html。

3. 创建一个产品订购表单。用三个独立的 HTML 文档，每个文档包含不同的表单。第一个页面包含产品信息，第二个页面包含客户订购信息，第三个页面包含运输信息。可以允许用户在三个页面之间向前和向后浏览。当用户在页面之间切换时，将用户输入的信息保存到表单隐藏域中。程序中还要包括对要求输入域进行校验的 JavaScript 代码。给文件取名并保存文件到 Tutorial.06 文件夹。

4. 创建一个表单，允许用户为一个专业会议签名。并将表单提交到您的 E-mail 地址。保存文件到 Tutorial.06 文件夹，文件名称为 Conference.html。

5. 为您的聚会创建一个 RSVP 表单。您的客人将填写这些表单并提交到您的 E-mail 地址。保存文件到 Tutorial.06 文件夹，文件名称为 RSVP.html。

6. 创建一个表单，处理雇员的度假申请。用 JavaScript 来检查职员必须填写的信息，诸如他们计划的假期日期。将表单的提交按钮命名为 Submit Request。找一个可以扮演您的上级的人，然后将该表单提交到他的 E-mail 地址。保存文件到 Tutorial.06 文件夹，文件名称为 VacationRequest.html。

第 7 章　动态 HTML 和动画

案例

越来越多的 WebAdventure 的客户询问他们的网站是否能够包含格式和图像，在更新它们时不需要重新从服务器载入 HTML 文档。他们还想寻找新的方法来使用动画和实现交互行为以便吸引和使客户流连忘返，并且使他们的网站令人印象深刻和便于浏览。尽管标准的 HTML 不能够实现这些类型的效果，但是您的经理已经要求您学会怎样使用动态的 HTML（DHTML）来满足客户的需求。如 Rocking Horse Toys, Inc.客户想在他们的网页中嵌入驰骋的骏马来作为公司的图标。经理还想让您利用 DHTML 技能来为 WebAdventure 创建活泼生动的图标。

浏览动画

动态 HTML 允许在浏览器演示网页后更改 HTML 标签。动态演示 HTML 标签的能力使您能够为网页中的其他内容添加动画。本辅导通过动画来学习动态 HTML 技术。在学习动态 HTML 过程中，将创建两个不同的动画：驰骋的骏马和绕太阳旋转的地球。两个动画实例能够用于广告或产生可视化效果来吸引访问者浏览网页。

浏览驰骋的骏马动画：

1．在浏览器中，从 Student Disk 上的 Tutorial.07 文件夹中打开 Tutorial7_RockingHorse.html 文件。图 7-1 显示了在 Navigator 中浏览 Tutorial7_RockingHorse.html 的实例的结果。

2．单击“Start Rocking”按钮运行动画。在观看完动画之后，单击“Stop Rocking”按钮。

3．关闭浏览器窗口。

4．下一步，在文本编辑器或 HTML 编辑器中查看 Tutorial7_RockingHorse.html 文件的代码。<SCRIPT>部分包含了在动画中使用的的全局变量。使用 rockHorse()函数（在 startRocking()函数中调用）来运行动画。文档体内的按钮使用 onClick 方法来调用 startRocking()函数。

5．在查看完代码之后关闭文本编辑器或 HTML 编辑器。

下一步，将浏览轨道动画。

浏览轨道动画：

1．在浏览器中，从 Student Disk 的 Tutorial.07 文件夹打开 Tutorial7_OrbitMaster.html 文件。注意在 Internet Explorer 中，Turtorial7_OrbitMaster.html 文件打开的是 Tutorial7_OrbitIE.html，而在 Navigator 中，打开的是 Tutorial7_OribitNavigator.html。图 7-2 显示了在 Navigator 中浏览轨道动画实例的结果。

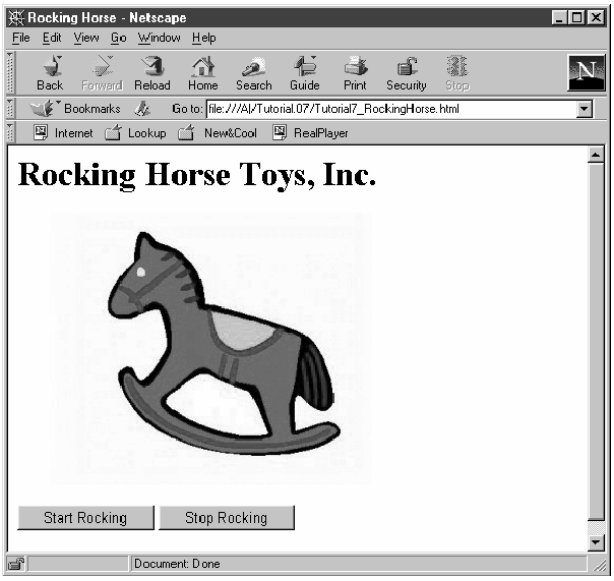


图 7-1　浏览器中 Tutorial7_RockingHorse.html

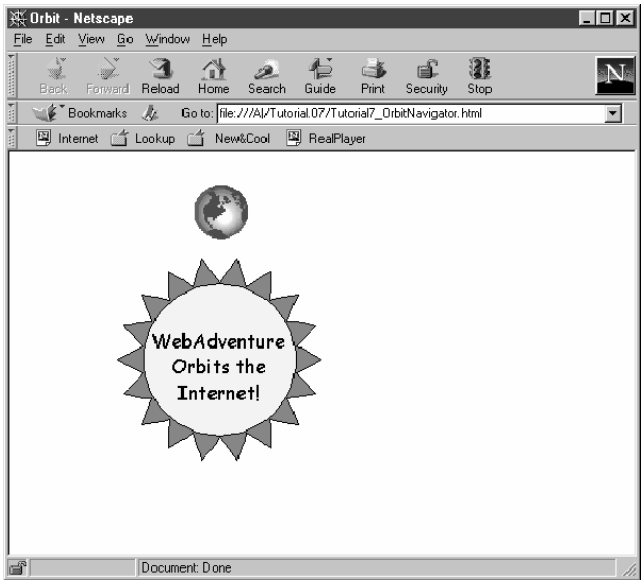


图 7-2　Navigator 中的轨道动画

2．关闭浏览器窗口。

3．在文本编辑器或 HTML 编辑器中查看 Tutorial7_OrbitMaster.html 文件的代码。在

<BODY>标签内的 onLoad 事件调用了 checkBrowser()函数，通过它来打开不同版本的动画文件，即 Tutorial7_OrbitIE.html 或 Tutorial7_OrbitNavigator.html，选中谁取决于运行的是 Internet Explorer 还是 Navigator。

4．下一步，比较 Tutorial7_OrbitIE.html 和 Tutorial7_OrbitNavigator.html 文件中的代码。两个文件都包含了 orbit()函数。注意在每个文件声明处的区别。还请注意每个文件都使用了不同的 HTML 标签。7.2 节将解释为什么 Internet Explorer 和 Navigator 对动态的 HTML 文件需要不同的 JavaScript 语法和 HTML 标签。

7.1 动态对象模型

本节目标

在本节将学习：

- ✧ 动态 HTML
- ✧ 文档对象模型
- ✧ Document 对象属性
- ✧ Document 对象方法
- ✧ Image 对象
- ✧ 利用 Image 对象实现动画
- ✧ 图像缓冲

7.1.1 动态 HTML

使用 Web 有多种目的。研究人员、科学家和学院里的人们使用 Web 来显示、搜索和查找信息。Web 的商业应用程序包括出版、广告和娱乐。今天多数企业都拥有自己的网站，并且将来可能把业务放在 Web 上。为了吸引和使访问者流连忘返，网站必须令人激动和充满视觉刺激。特别是，企业想让他们的网站包括广告、动画、与用户的交互、直观地浏览控件以及多种其他类型的效果，它们能够有助于销售他们的产品。

除了商业网站，还存在许多不同类型的娱乐网站，包括游戏、动画和多媒体演示。动画和多媒体演示需要快速地装载以便维持用户的兴趣。游戏必须是令人激动的并且是可交互的。例如，提供象棋游戏的网站必须让访问者能够移动每个棋子。

若 HTML 是动态的，那么它将更加有价值。在 Internet 术语中，动态具有几层含义。首先，“动态”指网页能够通过按钮或其他类型的控件来响应用户的请求。例如，动态网页可以允许用户改变文档的背景颜色，在用户提交表单时处理请求或通过其他方式与用户交互，如在线游戏或测验。动态还可表示不同类型的效果，它们能够在浏览器中自动显示，

如动画。

通过超文本链接，能够模拟有限的动态和交互。考虑一下图 7-3 中显示的网页，它显示了 WebAdventure 内编程人员的照片。此网页链接了其他 7 个网页，除了照片不同，其他部分是相同的。图 7-4 显示了 HTML 文件中用来显示 Erica Miller 网页的标签。

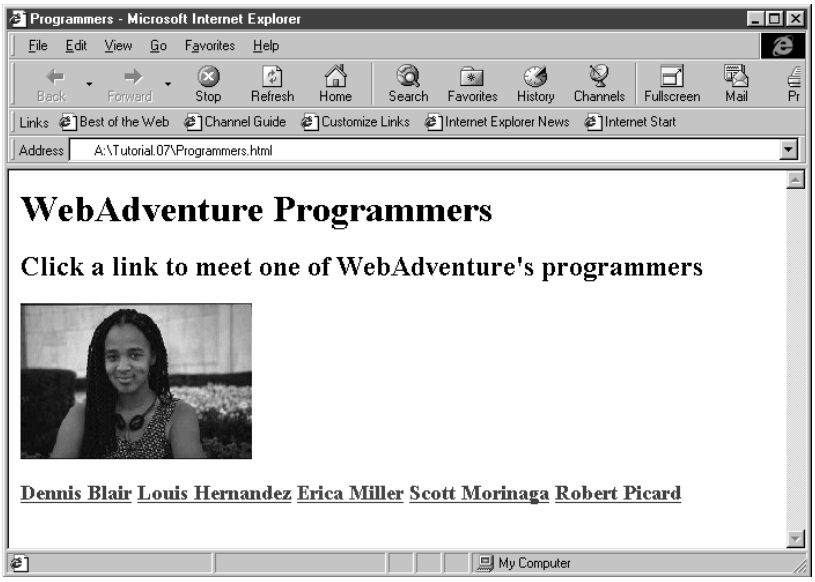


图 7-3　Programmers.html

```
<H2>Click a link to meet one of
WebAdventure’s programmers</H2>
<IMG NAME="programmer_pic" SRC="Miller.jpg"><P>
<H3>
<A HREF="Blair.html">Dennis Blair</A>
<A HREF="Hernandez.html">Louis Hernandez</A>
<A HREF="Miller.html">Erica Miller</A>
<A HREF="Morinaga.html">Scott Morinaga</A>
<A HREF="Picard.html">Raymond Picard</A>
</H3>
</BODY>
</HTML>
```

图 7-4　Miller.html

超文本链接并没有修改当前显示的文档，它们只是从服务器载入了新的文档，因此它们并不能创建真正的动态效果。当用户单击 Programmers 网页中的链接时，看起来好像仅改变了图像，实际上是替换了整个网页。浏览器不得不在服务器上查找正确的页面，将文件传递给计算机，同时显示新的文档。在此简单的实例中，您可能感觉不到完成这些步骤所花费的时间，然而对那些大而且复杂的页面，花费在传输和重新显示上的时间将是很长

的。如果此页面是动态的，那么仅需要改变<IMG>标签所显示的文件，并且由浏览器（而不是由服务器）在本地完成此任务。仅改变图像可能令人印象更深刻和更有效。

结合多种 Internet 技术（称为动态 HTML 或 DHTML）可以动态地改变 HTML 标签和显示页面。DHTML 用来指为 HTML 页面提供动态能力不同技术的结合。用来创建 DHTML 的技术包括 JavaScript、VBScript、CGI、Active Server Pages 以及其他技术。对我们来说，DHTML 包括下列技术：

- ✧ JavaScript
- ✧ HTML
- ✧ 层叠式表单

提示：在学习本教程时，请记住 DHTML 不是指单一的一种技术，而是多种技术的结合。

提示：层叠式表单用来管理 HTML 文档的格式化内容，在 7.2 节将学习层叠式表单。

图 7-5 显示了两类流行的 DHTML 的特点：滑动式菜单（在用户将鼠标移动至它上面时才弹出）和马赛克文字（在屏幕上滚动的消息）。马赛克文字显示在浏览器窗口的底部，它就在 WebAdventure's Mission 文字的下面。

图 7-5　滑动式菜单和马赛克文字

图 7-6 显示了用 DHTML 编写的游戏。它是著名的将动画（方块的移动）和交互（旋转和移动方块的键）结合起来的俄罗斯方块游戏。

提示：在图 7-6 中的 TetriScript 游戏是由日本的 Tochigi 的 Kazuhiro Moriyama 编写的。在 http://plaza.harmonix.ne.jp/~jimmeans/下还能够找到其他优秀的 JavaScript 游戏。

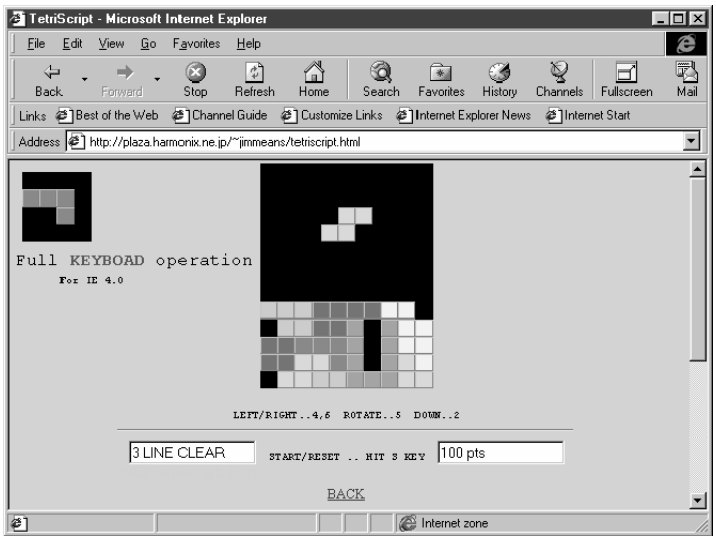


图 7-6　TetriScript 游戏

7.1.2　文档对象模型

DHTML 的核心是文档对象模型，如图 7-7 所示。文档对象模型（DOM）用来表示在窗口中显示的 HTML 文档，并且提供了访问文档元素的编程接口。DOM 实际上是第 5 章所学的 JavaScript 对象模型的 Document 对象分支。任何时候在使用图像和表单或引用文档对象时，其实已经使用了 DOM。您已经使用过 Document 对象的两个方法：write()和 writeln()。

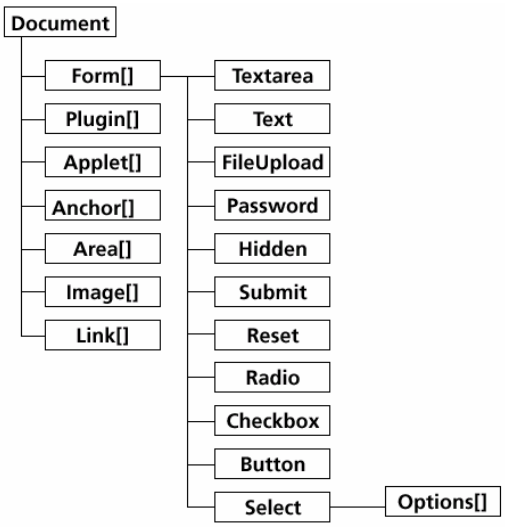


图 7-7　文档对象模型

提示：Document 对象不仅用来指 HTML 文档，还用来指在浏览器中显示的其他文件类型，如.jpg、.gif 和.xml。.jpg 和.gif 是两种图像文件格式。

提示：图 7-7 仅列出了 DOM 的 HTML 对象部分。DOM 还包括事件、属性和方法。在第 2 章学习了事件。DOM 的属性和方法将在本节内讨论。

DOM 将 HTML 文档中的每个标签转换成自己的编程对象以便 JavaScript 能够访问每个 HTML 元素。此功能让您在页面显示之后能够动态地改变每个 HTML 元素，而不需要重新从服务器载入页面。可以使用 NAME 属性或 Array 对象中元素号来引用 HTML 标签。下面的代码演示了一个简单的表单，在用户单击按钮时，它能够动态地改变在文本<INPUT>标签内显示的文本。图 7-8 显示了程序在浏览器中运行的结果。在 onClick 事件内，将 Form 对象和 elements[]数组与 this 引用一起使用。请注意代码未载入新的 HTML 文档或重新载入源文档，只是动态地修改了文本框的值。没有 DHTML，实现此种的功能是不可能的。

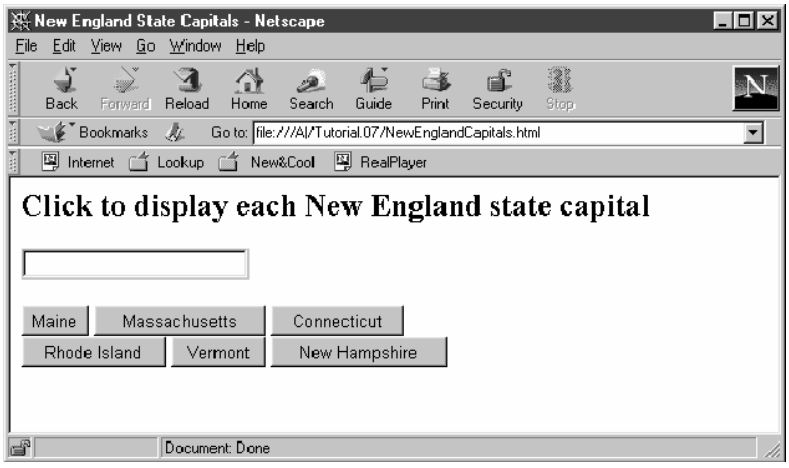


图 7-8　州首府程序

尽管组成 DHTML 的每种技术已被接纳为标准，但是 DHTML 的实现还在不断地发展。因为 DHTML 技术还在不断地发展，所以还没有已被接纳的 DOM 标准。Navigator 4 和 Internet Explorer 4 都支持 DOM 的最初版本——0 级。然而，Internet Explorer 4 的一些 DOM 对象完全不与 0 级兼容，因此也就不与 Navigator 4 兼容。Navigator 4 也包含了不兼容 0 级的对象，但是不如 Internet Explorer 4 那么多。在撰写此教程的时候，World Wide Web 协会（W3C）正在起草新的 DOM 标准——1 级。DOM 1 级将包括许多在 Internet Explorer 4 中已支持的对象、属性和方法。然而，在 Internet Explorer 5 和 Navigator 5 发布之前，DOM 1 级仍不可使用。即使在新版本的 Web 浏览器发布之后，让市场完全接受它还需时日。为了开发能够在多数正在使用的浏览器中运行 DHTML Web 页面，在 Internet Explorer 5 和 Navigator 5 广泛使用之前，应该仅使用 DOM 0 级对象。

提示：若可能，此教程仅介绍与 Navigator 4 和 Internet Explorer 4 兼容的对象、属性和方法。

Document 对象属性

Document 对象包含了用来使用 HTML 对象的各种属性。在表 7-1 中列出了 Document 对象的属性。

表 7-1　Document 对象属性

属　　性	描　　述
alinkColor	激活链接的颜色，由<BODY>标签的 ALINK 属性指定
anchors[]	引用文档的 anchors 的数组
applets[]	引用文档的 applets 的数组
bgColor	文档的背景颜色，由<BODY>标签的 BGCOLOR 属性指定
cookie	为当前文档指定 cookie
domain	当前文档所在的服务器的域名
embeds[]	引用文档插件和 ActiveX 控件的数组
fgColor	文档的前景文本颜色，由<BODY>标签的 FGCOLOR 指定
forms[]	引用文档表单的数组
images[]	引用文档图像的数组
lastModified	文档最后被修改的日期
linkColor	文档中的未浏览的链接的颜色，由<BODY>标签的 LINK 属性指定
links[]	引用文档的链接的数组
referrer	文档的 URL，它们提供了指向当前文档的链接
title	文档的标题，由在文档<HEAD>部分的<TITLE>…</TITLE>标签对指定
URL	当前文档的 URL
vlinkColor	文档的已浏览的链接的颜色，由<BODY>标签的 VLINK 属性指定

在 Navigator 4.0 中，大部分 Document 对象的属性仅能在显示文档前设置。在文档被显示之后，仅能够动态地设置 bgColor 属性。因此，在 0 级 DHTML 中，许多文档对象属性在创建动态文档方面作用有限。而用来检索与当前文档相关的信息的属性则更有用。例如，下面的函数在警告对话框中显示了文档的标题、URL 和最后被修改的日期。

```
function documentStatistics() {
    alert(document.title + "\n" + document.URL + "\n"
        + document.lastModified);
}
```

与 Navigator 4.0 相比，Internet Explorer 4.0 允许在显示文档之后动态地改变许多 Document 对象属性。请记住动态改变 Document 对象属性的与 Internet Explorer 兼容的文档在 Navigator 中将不能运行。在 Navigator 中，在显示文档之后只能够动态地设置 bgColor 属性。图 7-9 演示了一个用于 Internet Explorer 的程序实例，它使用 setInterval()方法每隔 1000 毫秒（1 秒）动态地改变文档的 bgColor 和 fgColor。通过在<BODY>标签内的 onLoad 事件

中调用 setInterval()方法来调用 changeColor()函数。实例创建了一种“闪烁的标志”，它能够用在 Web 网站的广告内。图 7-10 演示了程序的首次运行，它包含了以白为背景的黑色文本，在图 7-11 中演示了程序的第二次运行，它包含了以黑为背景的白色文本。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
var backColor = "black";
var textColor = "white";
function changeColor() {
        if (backColor == "black") {
                backColor = "white";
                textColor = "black";
                document.bgColor = backColor;
                document.fgColor = textColor
        }
        else {
                backColor = "black";
                textColor = "white";
                document.bgColor = backColor;
                document.fgColor = textColor
        }
}
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
<BODY onLoad="setInterval("changeColor();",1000);">
<H2>Choose WebAdventure for all your Internet needs!</H2>
</BODY>
```

图 7-9 改变 bgColor 和 fgColor 属性

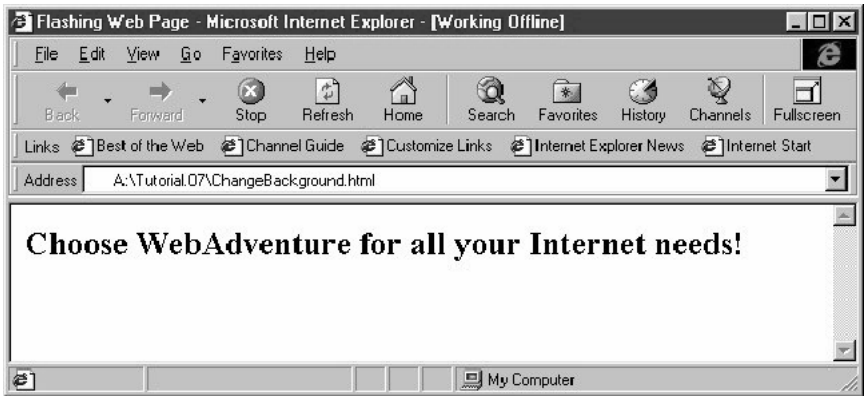


图 7-10 白色背景的黑色文本

图 7-11　黑色背景的白色文本

提示：请记住改变 bgColor 和 fgColor 的程序仅能在 Internet Explorer 4.0 或高版本中运行。

下一步将创建使用 setInterval()方法每隔数秒自动改变背景颜色的网页。目标是创建与 Internet Explorer 和 Navigator 都兼容的程序。既然在 Navigator 中不能设置 fgColor 属性，那么将不像前面的实例那样包括自动改变文本前景颜色的代码。

创建使用 setInterval()方法每隔数秒自动改变背景颜色的网页：

1．启动文本编辑器或 HTML 编辑器，创建新文档。

2．输入文档的<HTML>和<HEAD>开始部分。

```
<HTML>
<HEAD>
<TITLE>Changing Background</TITLE>
```

3．在<HEAD>段内为 JavaScript 添加起始语句。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

4．输入 var currentColor = "blue"; 创建保存当前所显示的颜色的变量。

5．输入下列函数，它们使用 Document 对象的 bgColor 属性来改变网页的背景。在<BODY>标签内使用 onLoad 事件来调用此函数。

```
function changeBackground(){
        if (currentColor == "blue") {
                document.bgColor = "red";
                currentColor = "red";
        }
        else {
```

```
            document.bgColor = "blue";
            currentColor = "blue";
        }
    }
```

6．添加下列标签来结束<SCRIPT>和<HEAD>段。

```
//STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
```

7．创建下面的包括了 onLoad 事件处理器<BODY>标签，它使用了 setInterval()方法来调用首部内的 changeBackground()函数：

```
<BODY onLoad="setInterval('changeBackground();',2000);">
```

8．输入下列行来解释网页的目的。

```
<B>The background of this Web page changes from
blue to red every two seconds.</B>
```

9．添加下列行来结束<BODY>和<HTML>标签。

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

10．将文件在 Student Disk 的 Tutorial.07 文件夹内保存为 ChangeBackground.html。在浏览器内打开 ChangeBackground.html 文件。背景应该每隔 2 秒就由蓝变为红。

11．关闭浏览器窗口。

Document 对象方法

在创建网页时，还想动态地创建新的 HTML 文档。例如，在运行 CGI 脚本处理在线订单之后，想动态地创建新的 HTML 文档来确认订单。文档对象包含了 4 个方法来动态地创建网页：write()、writeln()、open()和 close()。close()方法用来通知浏览器已经编写完窗口或框架，应该显示文档了。在整个教程中，已使用了 write()和 writeln()方法来为新的页面在显示时添加内容。open()方法打开新窗口或框架而不是当前窗口或框架，并使用 write()和 writeln()方法来更新它的内容。在 open()方法内能够包括参数，用来指定被显示的文档的 MIME 类型。若不包括参数，将使用 text/html 的默认的 MIME 类型。若使用 write()和 writeln()方法而不是用 open()方法，那么当前窗口的内容将被覆盖掉。

提示：在第 6 章学习了 MIME 类型。

write()和 writeln()方法的缺点在于在显示页面之后不能使用它们来修改内容。可以在显示当前文档之后在其内运行 write()和 writeln()方法，但是这样做将会覆盖已有的文档内容。尽管 write()和 writeln()方法不能用来动态地修改已有的页面，但是可以使用它们来动态地创建新窗口或框架。

众所周知，Document 对象的 write()和 writeln()方法需要文本字符串作为参数。write()和 writeln()之间的惟一的区别是 writeln()方法在行后添加了一个回车字符。为了让浏览器识别在 writeln()方法之后的分行，必须在<PRE>…</PRE>标签对内封装包含了 writeln()方法的<SCRIPT>…</SCRIPT>标签对。可以在 write()方法内的文本参数内包括换行字符（\n），在不使用<PRE>标签和 writeln()方法情况下进行分行。下面的代码演示了 write()和换行字符是怎样工作的：

```
<SCRIPT LANGUAGE="JavaScript1.2">
document.write("Hello World \n")
document.write("This line is printed
      below the 'Hello World' line.")
</SCRIPT>
```

可以将每个文本字符串作为参数（用逗号分隔）包括在 write()方法中来生成多行，而不是分别使用 write()和 writeln()方法。下面的代码实现与前面实例相同的功能，但是这次仅使用了一个 write()方法：

```
<SCRIPT LANGUAGE="JavaScript1.2">
document.write("Hello World \n", "This line is printed
      below the 'Hello World' line.")
</SCRIPT>
```

下面的代码创建了一个新窗口，在文档找不到或不能访问时，将它作为重定向窗口。此段代码打开新窗口，接着使用 write()方法来创建一条具有指向 Document 对象的 referrer 属性的链接的消息。referrer 属性返回已打开的当前 Web 页面的 URL。在此例中，使用 referrer 属性来生成返回源 URL 的链接。在执行完 write()方法之后，接着执行 close()方法并在新窗口内显示此页面。既然 Window 和 Document 对象都包含了 open()和 close()方法，那么代码包括了每个对象引用来区分它们。图 7-12 演示了新窗口显示的结果。

```
var message = "<H2>The Web page you requested is
      currently unavailable.</H2>"
var link = "Click <A HREF='" + document.referrer
      + "'>here</A> to return to the last Web page."
redirectWindow = window.open("");
redirectWindow.document.open("text/html");
redirectWindow.document.write(message, link);
```

```
redirectWindow.document.close();
```

图 7-12 重定向窗口

7.1.3 Image 对象

网页最令人赏心悦目部分之一是图像。若没有图像，那么网页将仅仅是文本和超文本链接的集合。现在网页内的图像类型包括企业图标、产品图像、动画、图像映射（如第 2 章所创建的）以及其他类型的图像。不包括图像的面向商业的网页将很难吸引和留住访问者。

在网页内包括图像并不是新的功能。只需在<IMG>标签中将 SRC 属性设置为所显示的图像的 URL。HTML 文档中的每个<IMG>标签由 DOM images[]数组内的 Image 对象来表示。Image 对象表示使用<IMG>标签所创建的图像。若想根据用户的选择来改变图像作为计时广告程序的一部分，或用于简单的动画，那么必须在 JavaScript 中使用 Image 对象。

表 7-2 列出了 Image 对象属性，表 7-3 列出了 Image 对象事件。

表 7-2 Image 对象属性

属　性	描　述
border	只读属性，指出了边界宽度所包含的像素个数，由<IMG>标签的 BORDER 属性指定
complete	布尔值，在完全载入图像时返回真
height	只读属性，指出了图像的高度，由<IMG>标签的 HEIGHT 属性指定
hspace	只读属性，指出了图像的左边到右边水平空白所包含的像素个数，由<IMG>标签的 HSPACE 属性指定
lowsrc	以低分辨率显示的图像的 URL
name	赋给标签的名称

续表

属　性	描　述
src	所显示的图像的 URL
vspace	只读属性，指出了图像的上部和底部垂直空白所包含的像素个数，由<IMG>标签的 VSPACE 属性指定
width	只读属性，指出了图像的宽度，由<IMG>标签的 WIDTH 属性指定

表 7-3　Image 对象事件

事　件	描　述
onLoad	在载入图像之后运行
onAbort	在用户单击停止按钮取消载入图像时运行
onError	在载入图像的过程中出现错误时运行

Image 对象最重要的元素之一是 src 属性，它允许 JavaScript 动态地改变图像。

提示：src 和 lowsrc 是 Navigator 显示文档之后惟一能够被改变的 Image 属性。Internet Explorer 允许动态地改变大部分其他属性。

图 7-13 对图 7-4 中相同程序进行了改进，并介绍了怎样使用 src 属性。在此例中，每个程序员的超文本链接被表单按钮所代替。最重要的是，在用户单击不同的程序员时不再替换 HTML 文档。相反地，通过 Image 对象的 src 属性动态地替换了由<IMG>标签所显示的图像。新程序比每次想显示新的图像时载入新的 HTML 文档（如前面的实例）效率更高。程序显示的默认图像是列表中的第一个程序员——Dennis Blair。图 7-14 显示了在浏览器中的结果。

下一步，将创建让用户动态地改变<IMG>标签所显示的图像的文档。将创建三个按钮，用它们来显示同一幅图像的不同尺寸的图像。文档中所使用的图像位于 Student Disk 的 Tutorial.07 文件夹内。

创建让用户动态地改变<IMG>标签所显示的图像的文档：

1．启动文本编辑器或 HTML 编辑器，并创建新文档。

2．输入文档的<HTML>和<HEAD>段。

```
<HTML>
<HEAD>
<TITLE>Image Options</TITLE>
</HEAD>
```

3．添加<BODY>体的起始部分，接着输入<FORM>添加表单。

4．创建下面的<INPUT>标签，它们将分别显示鸟的小图像、中等图像或大图像。标签使用 onClick 事件处理器来改变 Image 对象的高度和宽度。

```
<HTML>
<HEAD>
<TITLE>Programmers</TITLE>
</HEAD>
<BODY>
<H1>WebAdventure Programmers</H1>
<H2>Click a button to meet one of
WebAdventure's programmers</H2>
<IMG NAME="programmer_pic" SRC="blair.jpg"><P>
<FORM NAME="programmer_list">
    <INPUT TYPE="button" VALUE="Dennis Blair"
        onClick="document.programmer_pic.src='blair.jpg'">
    <INPUT TYPE="button" VALUE="Louis Hernandez"
        onClick="document.programmer_pic.src='hernandez.jpg'">
    <INPUT TYPE="button" VALUE="Erica Miller"
        onClick="document.programmer_pic.src='miller.jpg'">
    <INPUT TYPE="button" VALUE="Scott Morinaga"
        onClick="document.programmer_pic.src='morinaga.jpg'">
    <INPUT TYPE="button" VALUE="Raymond Picard"
        onClick="document.programmer_pic.src='picard.jpg'">
</FORM>
</H3>
</BODY>
```

图 7-13　Programmers.html

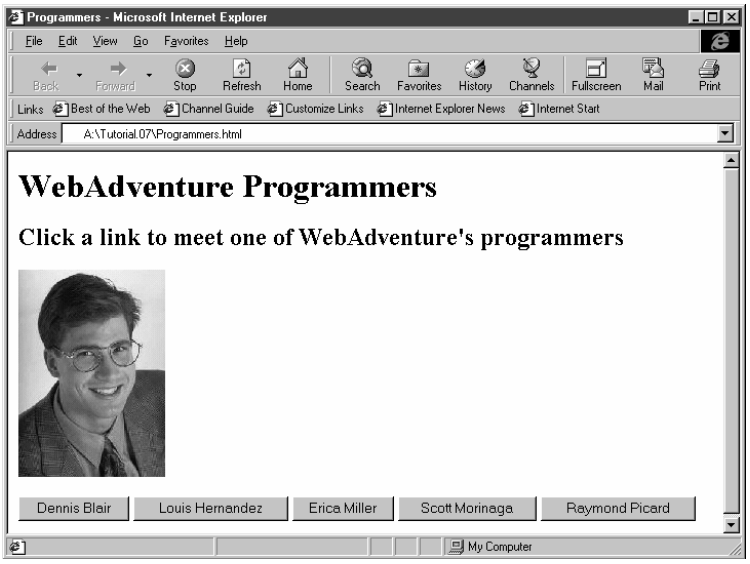


图 7-14　浏览器中的 Programmers.html

```
<INPUT TYPE="button"VALUE="Small Bird "
        onClick="document.bird.src='smallbird.gif';">
<INPUT TYPE="button" VALUE=" Medium Bird "
        onClick="document.bird.src='mediumbird.gif';">
<INPUT TYPE="button" VALUE=" Big Bird "
        onClick="document.bird.src='largebird.gif';">
```

5．输入</FORM>结束标签。

6．接着增加下面的添加<IMG>标签一行。它将图像初始化为 smallbird.gif。

```
<IMG SRC="smallbird.gif" NAME="bird">
```

7．输入</BODY>和</HTML>结束标签。

8．在 Student Disk 的 Tutorial.07 文件夹内将文件保存为 ChangeImage.html。在 Web 浏览器内打开 ChangeImage.html 文件。图 7-15 演示了在用户单击“ Medium Bird ”按钮之后在浏览器中文档所显示的结果。

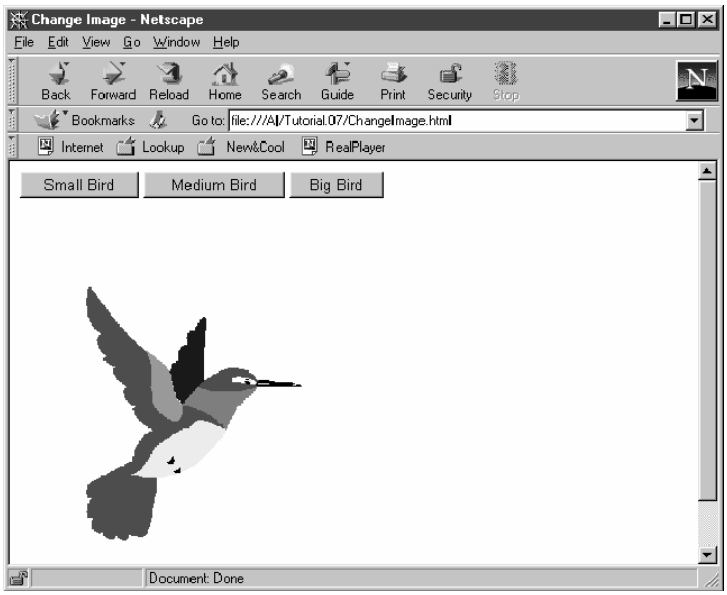


图 7-15　ChangeImage.html

9．关闭 Web 浏览器窗口。

7.1.4　使用 Image 对象的动画

在第 5 章中，学习了怎样使用 setTimeout()和 setInterval()方法来自动执行 JavaScript 代

码。通过将 Image 对象的 src 属性和 setTimeout()或 setInterval()方法结合起来，能够在 HTML 文档内创建简单的动画。此处动画并不具备卡通特征，但是它能够自动改变一系列图像。Web 动画还包括传统动画，如卡通和运动。实现动画的 JavaScript 程序实例包括了一个简单的广告（其中包括 2 幅图像，每 2 秒交替变换一次）和一个在线时钟的秒针（时钟的每个位置都需要一幅单独的图像）。图 7-16 的程序使用了 setInterval()方法每 2 秒就自动地切换广告图像。图 7-17 显示了这 2 幅图像。

```
<HTML>
<HEAD>
<TITLE>Advertisement</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
var qa = "q";
function changeImage() {
      if (qa == "q") {
                  document.animation.src = "answer.jpg";
                  qa = "a";
                  }
      else {
                  document.animation.src = "question.jpg";
                  qa = "q";
            }
}
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
```

图 7-16　变换图像的程序

图 7-17　广告图像

提示：若想查看闪烁广告是怎样运行的，在 Student Disk 的 Tutorial.09 文件夹内包括了 Advertisement.html 文件。

真正的涉及到运动的动画对人物或对象所作出的每个动作都需要使用不同的图像或帧。本教程并不教创建动画序列中帧所必需的绘画技能。相反，您的目标是怎样使用 JavaScript 和 Image 对象通过切换帧（通过<IMG>标签显示）来实现简单的动画。

提示：不要将动画帧和使用<FRAMESET>和<FRAME>标签创建的帧相混淆。

图 7-18 显示了 6 帧图像，每帧都包含了运动员奔跑的动作，它们作为动画序列中的帧的实例。

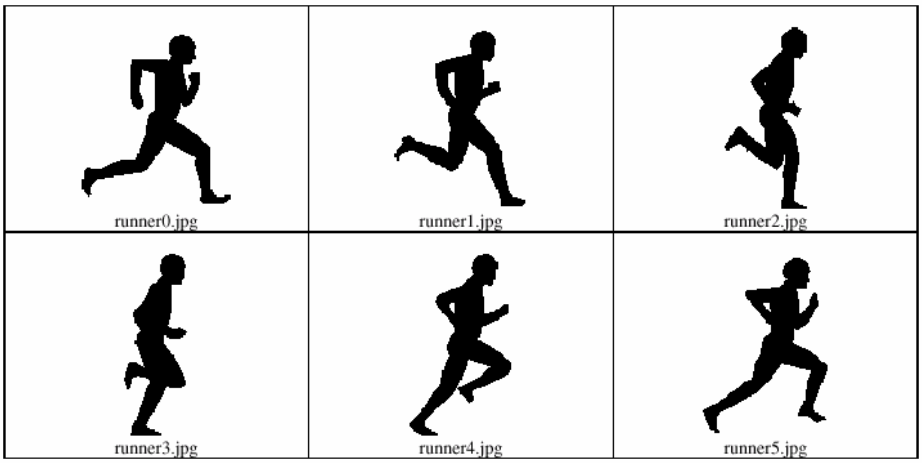


图 7-18　动画帧

通过使用 JavaScript 中的 setInterval()或 setTimeout()方法创建动画序列来循环动画序列中的帧。setInterval()或 setTimeout()方法的每次循环都改变<IMG>标签所显示的帧。动画的速率取决于作为参数传入 setInterval()或 setTimeout()方法的毫秒数。

在创建动画序列时，为了保证序列中的所有图像都以相同的尺寸显示，确保动画的每帧都是用相同的<IMG>标签的 HEIGHT 和 WIDTH 属性创建是十分重要的。

图 7-19 的代码使图 7-18 中的帧运动起来。代码将帧赋给 runner[]数组。在单击“ Run ”按钮之后，setInterval()方法执行 if 语句，它根据 curRunner 变量改变所显示的帧。一旦 curRunner 值为 5（具有 6 个元素的数组的最大下标），它立即将重置为 0（数组中的第一个元素），接着动画序列从头开始。显示在图 7-18 中帧的每幅图像的名称与 runner[]数组中的元素索引相对应。“ Stop ”按钮使用 clearInterval()方法来停止执行 startInterval()方法。图 7-20 演示了程序在浏览器中的结果。

提示：若想查看 runner 动画是怎样运行的，在 Student Disk 的 Tutorial.09 文件夹内包括了 Runner.html 文件。

```
<HTML>
<HEAD>
<TITLE>Runners</TITLE>
<SCRIPTLANGUAGE="JavaScript1.2">
<!--HIDEFROMINCOMPATIBLEBROWSERS
var runner=newArray(6);
var curRunner=0;
var startRunning;
runner [0 ]="runner0.jpg";
runner [1 ]="runner1.jpg";
runner [2 ]="runner2.jpg";
runner [3 ]="runner3.jpg";
runner [4 ]="runner4.jpg";
runner [5 ]="runner5.jpg";
functionmarathon(){
    if(curRunner==5)
      curRunner=0;
    else
      ++curRunner;
    document.animation.src=runner [curRunner ];
}
//STOP HIDING FROM INCOMPATIBLE BROWSERS-->
</SCRIPT>
</HEAD>
<BODY>
<IMG SRC="runner0.jpg"NAME="animation"><P>
<FORM>
<INPUT TYPE="button"NAME="run"VALUE="Run"
    onClick="startRunning=setInterval(‘marathon()',100);">
<INPUT TYPE="button"NAME="stop"VALUE="Stop"
    onClick="clearInterval(startRunning);">
</FORM>
</BODY>
</HTML>
```

图 7-19　runner 动画代码

尽管 runner 动画为图像增添了动态特性，但是它还存在缺点：动画不能在屏幕上移动，而是限制在单个<IMG>标签所占的位置内运动。runner 是运动的，但是给人的印象是它只是在固定的位置上运动。在下一节，将学习怎样使用层叠式表单来改变图像在屏幕上的位置，因此动画将是真正地在运动。

下面将创建驰骋的骏马动画。动画所需的 6 幅图像位于 Student Disk 的 Tutorial.07 文件夹内。图 7-21 显示了所有图像。

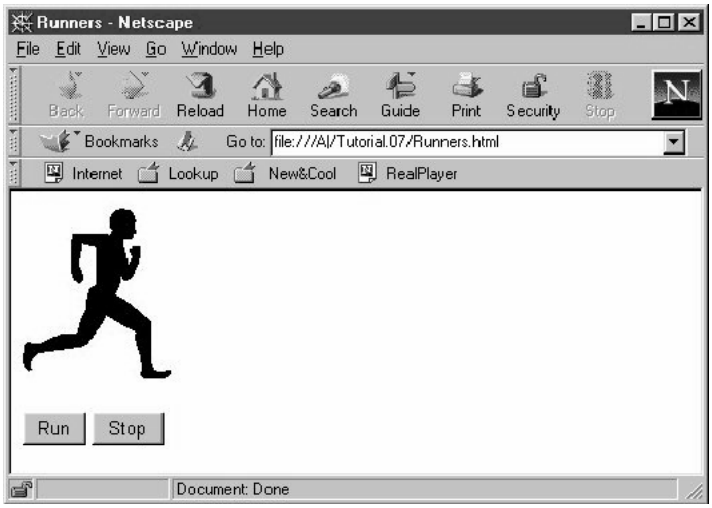


图 7-20　浏览器中的 runner 动画

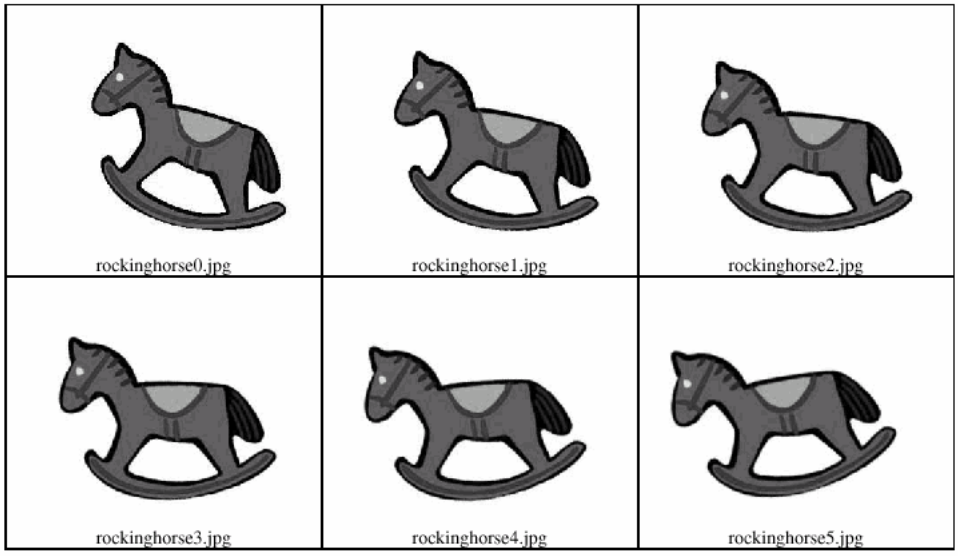


图 7-21　驰骋的骏马帧

驰骋的骏马动画与 runner 动画类似。然而，与 runner 动画不同的是，在显示完最后一帧——rockinghorse5.jpg 之后，驰骋的骏马动画不是重新显示首帧——rockinghorse0.jpg。若重新显示首帧，动画将不能平滑地运行。驰骋的骏马动画在显示最后一帧之后必须以相反的顺序循环每帧，而不是重新从首帧开始。

创建驰骋的骏马动画：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．输入文档的<HTML>和<HEAD>段。

```
<HTML>
```

```
<HEAD>
<TITLE>Rocking Horse</TITLE>
```

3．为 JavaScript 段添加起始语句。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

4．输入下面 4 个变量。horses 变量将包含 6 幅图像的文件名。curHorse 变量将作为计数器在 if 语句内使用。direction 变量将用来确定马是从右边还是从左边驰骋。begin 变量将赋给启动动画的 setInterval()方法。

```
var horses =new Array(5);
var curHorse = 0;
var direction;
var begin;
```

5．将图像文件赋给 horses 数组中相应的元素。

```
horses[0] = "rockinghorse0.jpg";
horses[1] = "rockinghorse1.jpg";
horses[2] = "rockinghorse2.jpg";
horses[3] = "rockinghorse3.jpg";
horses[4] = "rockinghorse4.jpg";
horses[5] = "rockinghorse5.jpg";
```

6．下面的函数实现了动画。第一条 if 语句判断 curHorse 变量等于 0 还是等于 5。如果变量等于 0，那么 direction 变量设置为 right，如果变量等于 5，那么 direction 变量设置为 left。下一条 if 语句接着判断 direction 变量来确定是递增还是递减 curHorse 变量。在函数内的最后一条语句将名为 animation 的<IMG>标签所显示的图像改变为与 curHorse 变量相匹配的 horses[]数组内相应的元素。在第 9 步将创建动画<IMG>标签。

```
function rockHorse(){
    if (curHorse == 0)
        direction = "right";
    else if (curHorse == 5)
        direction = "left";
    if (direction == "right")
        ++curHorse;
    else if (direction == "left")
        --curHorse;
```

```
        document.animation.src = horses[curHorse];
}
```

7．下一步，添加下面的函数，文档体内的“ Start Rocking ”按钮将调用它。若它正在运行，那么第一条语句使用 clearInterval()方法来终止动画。若不包括此语句，那么接着用户可能单击“ Start Rocking ”按钮数次，这将导致创建多个 setInterval()方法的实例。同一个 setInterval()方法的多个实例将导致计算机执行与 setInterval()方法实例同样多的动画序列，这样将导致动画不按要求的速率运行或无规律地运行。函数内的最后一条语句是 setInterval()方法，它将调用在第 6 步创建的 rockHorse()函数。

```
function startRocking(){
        clearInterval(begin);
        begin = setInterval("rockHorse()",100);
}
```

8．添加下面的代码来结束<SCRIPT>和<HEAD>标签。

```
//STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
```

9．添加下面的文档体段。名为 animation 的<IMG>标签将打开动画内的首帧——rockinghorse0.jpg。“ Start Rocking ”按钮使用 onClick 事件处理器来运行 startRocking()函数，接着“ Stop Rocking ”按钮清除 startRocking()函数所调用的 setInterval()方法。

```
<BODY>
<H1>Rocking Horse Toys, Inc.</H1><P>
<IMG SRC="rockinghorse0.jpg" NAME="animation"><P>
<FORM>
<INPUT TYPE="button" NAME="run" VALUE=" Start Rocking "
        onClick="startRocking()">
<INPUT TYPE="button" NAME="stop" VALUE=" Stop Rocking "
        onClick="clearInterval(begin)">
</FORM>
</BODY>
```

10．输入</HTML>结束标签。

11．在 Student Disk 的 Tutorial.07 文件夹内将文件保存为 RockingHorse.html。在浏览器内打开 RockingHorse.html 文件，接着查看它运行是否正常。

12．关闭 Web 浏览器窗口。

7.1.5 图像缓冲

在驰骋的骏马程序中，您可能已经注意到每幅图像的载入看起来不连续、无规律或十分缓慢。产生这种情况是因为 JavaScript 未在内存中保存能够在任何时候需要使用的图像。相反，每次通过<IMG>载入不同的图像时，JavaScript 必须从原图像文件重新打开或重新加载图像。可能直接从本地计算机上 Student Disk 来访问驰骋的骏马图像文件，或是运算速度十分快的计算机可能注意不到载入问题。若遇到无规则地载入图像，或是每次不得不从 Web 服务器载入它们，那么可以想象动画看起来将是多么地混乱和缓慢。一种用来消除多次下载相同文件的技术称为图像缓冲。图像缓冲在本地计算机内存中暂时地保存图像文件。此技术允许 JavaScript 在内存中保存和从内存中检索图像，而不是每次在需要它时下载图像。

使用 Image 对象的 Image()构造函数来缓冲图像。Image()构造函数创建新的 Image 对象。在 JavaScript 内缓冲图像需要三步：

- ✧ 使用 Image()构造函数创建新的对象
- ✧ 将图像文件赋给新的 Image 对象的 src 属性
- ✧ 将新的 Image 对象的 src 属性赋给<IMG>标签的 SRC 属性

在下面的代码中，名为 myImage 的<IMG>标签的 SRC 属性初始化为空串""。在<SCRIPT>段，将创建名为 newImage 的新的 Image 对象。newImage 对象用来保存和访问包含图像文件的内存缓冲。名为 graphic.jpg 的文件被赋给 newImage 对象的 src 属性。newImage 对象的 src 属性接着被赋给<IMG>标签的 src 属性。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
newImage = new Image()
newImage.src = "graphic.jpg"
document.myImage.src = newImage.src
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
<BODY>
<IMG NAME="myImage" SRC="first_graphic.jpg">
</BODY>
```

一定要理解在前面的代码中，graphic.jpg 文件并未直接赋给<IMG>标签的 SRC 属性。相反，newImage 对象被赋给了<IMG>标签的 SRC 属性。若使用语句 document.myImage.src = "graphic.jpg"直接将 graphic.jpg 文件赋给<IMG>标签的 SRC 属性，那么每次在需要它时都将重新加载文件。newImage 对象只打开文件一次并将它保存到内存缓冲中。

图 7-22 演示了使用图像缓冲改进后的 runner 动画代码。将每个图像文件添加给 runner[]

数组的代码行用 for 循环替换了。如在第 4 章中所学的，for 语句被用来重复一条语句或一系列语句，只要给定的条件表达式为真。图 7-22 中的 for 循环将新对象赋给 runner[]数组的每个元素直到计数器 i 大于 6，它代表数组的最大下标。runner[]数组中的每个对象接着使用 src 属性将图像文件赋给它。在 marathon()函数中，在语句 document.animation.src = runner[curRunner];中的 runner[curRunner]操作符现在包括了 src 属性以便语句读取 document.animation.src = runner[curRunner].src;。

```
<HTML>
<HEAD>
<TITLE>Runners</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
var runner = new Array(6);
var curRunner = 0;
var startRunning;
for(var i = 0; i < 6; ++i) {
        runner[i] = new Image();
        runner[i].src = "runner" + i + ".jpg";
}
function marathon() {
        if (curRunner == 5)
                curRunner = 0;
        else
                ++curRunner;
        document.animation.src = runner[curRunner].src;
}
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
<BODY>
<IMG SRC="runner1.jpg" NAME="animation"><P>
<FORM>
<INPUT TYPE="button" NAME="run" VALUE=" Run "
onClick="startRunning=setInterval('marathon()',100);">
<INPUT TYPE="button" NAME="stop" VALUE=" Stop "
            onClick="clearInterval(startRunning);">
</FORM>
</BODY>
</HTML>
```

图 7-22　在添加缓冲之后的 runner 动画代码

下一步，修改驰骋的骏马程序来添加图像缓冲。

为 RockingHorse.html 文档添加图像缓冲：

1．返回到文本编辑器或 HTML 编辑器内 RockingHorse.html 文件，接着在 Student Disk 的 Tutorial.07 文件夹内将它另存为 RockingHorseCache.html 新文件。

2．搜索下面的将每个驰骋的骏马帧赋给 horses[]数组的 6 条语句。

```
horses[0] = "rockinghorse0.jpg";
horses[1] = "rockinghorse1.jpg";
horses[2] = "rockinghorse2.jpg";
horses[3] = "rockinghorse3.jpg";
horses[4] = "rockinghorse4.jpg";
horses[5] = "rockinghorse5.jpg";
```

3．使用下面的 for 语句来替换前面的语句。for 语句创建了 horses[]数组每个元素内的新 Image 对象。接着使用 src 属性将图像文件赋给 horses[]数组内的每个对象。

```
for(var i =0;i <6;++i){
        horses[i] = new Image();
        horses[i].src = "rockinghorse" + i + ".jpg";
}
```

4．在 rockHorse()函数内为 document.animation.src = horses[curHorse];语句添加 src 属性以便它能够读取 document.animation.src = horses[curHorse].src;。

5．保存文档并在浏览器内打开文件。若前面看到的是混乱的动画，那么新动画看起来将十分平滑。

6．关闭 Web 浏览器窗口。

即使使用图像缓冲，在动画正常运行之前，图像必须全部被载入 Image 对象。通常，希望在载入页面后就立即开始运行动画。驰骋的骏马程序设计是载入图像对象后立即开始运行动画。然而，尽管载入了页面，但是所有的图像可能还未下载完也不可能被保存在内存中。若通过 Internet 连接来运行驰骋的骏马程序，onLoad 事件处理器可能在所有帧被传送和被赋给 Image 对象（取决于 Internet 连接速度）之前执行动画序列。动画仍能运行，但是在所有图像被成功地保存在 Image 对象之前，动画将十分混乱。为了确保在开始动画序列之前所有的图像已被下载到缓冲区，请使用图像对象的 onLoad 事件处理器。

图 7-23 演示了 runner 程序的另一个改进程序。这次程序不包括“ Run ”或“ Stop ”按钮。相反，<SCRIPT>段的 for 循环执行每个 Image 对象的 onLoad 事件。首先，runner[]数组的每个元素被赋给 runner 动画图像的相应的帧。语句 runner[i].onLoad = runMarathon(); 将 runMarathon()函数赋给每个 Image 对象的 onLoad 事件。在赋给 runner[]数组的每个 Image 对象的图像完成下载之后，它的 onLoad 事件将执行 runMarathon()函数，每次在一幅图像载入 runner[]数组后，imagesLoaded 变量递增 1。一旦 imagesLoaded 变量等于 6（表示所有

图像已被下载），marathon()函数执行调用 setInterval()的语句（最初位于“ Run ”按钮的 onClick 事件内）。

```
<HTML>
<HEAD>
<TITLE>Runner</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
var runner = new Array(6);
var curRunner = 0;
var startRunning;
var imagesLoaded = 0;
for(var i = 0; i < 6; ++i) {
      runner[i] = new Image();
      runner[i].src = "runner" + i + ".jpg";
      runner[i].onload = runMarathon();
}
function runMarathon(){
       ++imagesLoaded;
       if (imagesLoaded ==6)
             startRunning=setInterval("marathon()",100);
}
function marathon(){
       if (curRunner ==5)
             curRunner =0;
       else
             ++curRunner;
       document.animation.src =runner[curRunner].src;
}
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
<BODY>
<IMG SRC="runner1.jpg"NAME="animation"><P>
</BODY>
</HTML>
```

图 7-23　添加 onLoad 事件之后的 runner 动画代码

下一步，为驰骋的骏马程序添加图像 onLoad 事件，它在载入所有的图像之后执行动画。还将修改程序以便在载入所有图像之后立即执行动画。

为驰骋的骏马程序添加图像 onLoad 事件，并对它进行修改以便在载入所有图像之后立

即执行动画：

1．返回到文本编辑器或 HTML 编辑器内的 RockingHorseCache.html 文件，接着在 Student Disk 的 Tutorial.07 文件夹内将它另存为 RockingHorseImageLoaded.html 新文件。

2．在 var begin;语句后添加 var imageLoaded = 9;。imageLoaded 跟踪载入的图像数。

3．在将每个驰骋的骏马的图像文件赋给 horses[]数组的 for 循环的“ } ”的前面添加语句 horses[i].onLoad = loadImages;。在载入每幅图像时，它的 onLoad 事件调用 loadImages()函数。

4．在 for 循环后，创建下面的 loadImages()函数，它从第 3 步添加的 horses[I].onLoad = loadImages;语句处被调用。每次它被调用时，函数递增 imagesLoaded 变量。在 imagesLoaded 变量等于 6 时，setInterval()方法执行 rockHorse()函数。

```
function loadImages(){
        ++imagesLoaded;
        if (imagesLoaded == 6)
              begin=setInterval("rockHorse()",100);
}
```

5．从<SCRIPT>段删除 startRocking()函数，接着从文档体内删除表单。

6．保存文档并在 Web 浏览器内打开它。动画应该在所有图像载入之后立即执行。

7．关闭 Web 浏览器窗口。

在下一节，将学习怎样使用层叠式表单来创建更复杂的动画。

7.1.6 总结

- ✧ 动态 HTML（DHTML）结合了数种 Internet 技术，包括 JavaScript、HTML 和层叠式表单。它使 HTML 标签能够动态地改变，Web 页能够动态地显示，用户能够与 Web 页交互。
- ✧ 文档对象模型（DOM）代表在窗口内显示的 HTML 文档，它提供了用编程的方法来访问文档元素。
- ✧ Navigator 和 Internet Explorer 都支持 DOM 的最初版本—— 0 级。现在还没有已被接受的 DOM 标准。
- ✧ open()方法打开一个新窗口或框架而不是当前窗口或框架，为了更新它的内容，请使用 write()和 writeln()方法。
- ✧ close()方法通知 Web 浏览器已完成编写窗口或框架的内容，应该显示文档了。
- ✧ images[]数组包含了 HTML 文档内的所有图像，同样 forms[]数组包含了 HTML 文档内的所有表单。若 HTML 文档未包含任何图像，那么 images[]数组为空。
- ✧ Image 对象代表了使用<IMG>标签创建的图像。
- ✧ Image 对象的最重要的元素之一是 src 属性，它使 JavaScript 能够动态地改变图像。

- ✧ 通过将 Image 对象的 src 属性和 setTimeout()或 setInterval()方法结合起来，能够在 HTML 文档中创建简单的动画。
- ✧ 使用 JavaScript 的 setInterval()或 setTimeout()创建的动画序列被用来循环动画序列中的帧。
- ✧ 在内存中暂时保存图像文件的图像缓冲消除了多次下载相同的文件。此技术使 JavaScript 能够从内存中保存和检索图像而不是在每次需要它们时都下载图像。
- ✧ 使用 Image 对象的 onLoad 事件处理器来确保在所有图像下载到缓冲后才开始动画序列。

7.1.7　问题

1．下面哪种编程语言不能用来与用户交互？

a．Java

b．CGI

c．Perl

d．HTML

2．使用____来创建 DHTML。

a．JavaScript

b．HTML

c．层叠式表单

d．以上三种

3．Navigator 4 和 Internet Explorer 4 都支持哪种版本的 DOM？

a．0 级

b．1 级

c．2 级

d．都不是

4．____代表窗口内显示的 HTML 文档并提供了以编程的方式访问文档元素。

a．Window 对象

b．JavaScript

c．DOM

d．Frame 对象

5．在 Navigator 中，能够设置的 Document 对象的属性的时机是____。

a．仅在文档被显示之前

b．仅在文档被显示之后

c．在文档被显示之前和之后

d．在 Navigator 中不能设置 Document 对象属性

6．在 Internet Explorer 中，能够设置的 Document 对象的属性时机是____。

a．仅在文档被显示之前

b．仅在文档被显示之后

c．在文档被显示之前和之后

d．在 Internet Explorer 中不能设置文档对象属性

7．____方法打开新窗口或框架而不是当前窗口或框架，为了更新它的内容请使用 write()和 writeln()方法。

a．draw()

b．get()

c．update()

d．open()

8．____方法通知浏览器已编写完窗口或框架和文档可以被显示。

a．complete()

b．close()

c．update()

d．open()

9．____数组包含了所有的 HTML 文档的图像。

a．images[]

b．pictures[]

c．graphics[]

d．figures[]

10．在 JavaScript 中由____对象来表示 HTML 页的每个图像。

a．Picture

b．Graphic

c．Image

d．Figure

11．在 JavaScript 中使用 setInterval()和 setTimeout()方法和 Image 的____属性来创建简单的动画。

a．GIF

b．JPG

c．BMP

d．SRC

12．在 JavaScript 中动画的速度取决于____。

a．动画速率选项，在 Internet Explorer 的选项对话框或 Navigator 中的首选对话框中设置

b．计算机 CPU 的速度

c．动画序列内使用的帧数

d．作为参数传递给 setInterval()或 setTimeout()方法的秒数

13．下面的哪个步骤在图像缓冲中是不必要的？

a．使用 Image()构造函数创建新对象

b．将图像文件赋给新 Image 对象的 src 属性

c．将新 Image 对象的 src 属性赋给<IMG>标签的 src 属性

d．下载图像文件到本地硬盘

14．为确保在开始动画序列之前将所有的图像下载到了缓冲中，应使用 Image 对象的____。

a．onLoad 事件处理器

b．动画属性

c．loadImages()方法

d．images[]数组

7.1.8　练习

1．创建具有两个垂直框架的文档。在左框架中创建一系列的按钮，每个按钮代表所在州的名称。使用字典或百科全书来查找每个州的统计信息，如首府、行政官员的姓名、人口等。在用户单击州按钮时，使用 open()、close()和 writeln()方法在右框架中输出相应内容。在 Student Disk 的 Turtorial.07 文件夹内将文档存为 StateStatistics.html。

2．使用 Paint 或其他图像应用程序来创建一系列的帧，图中的棒作为起重器。在每帧内，根据起重器上升的阶段或距离来定位棒图像。使用 JavaScript 使它们动起来。在 Student Disk 的 Tutorial.07 文件夹内将文档存为 JumpingJack.html。

3．在 Student Disk 的 Tutorial.07 文件夹内包含了六幅风车图像：从 windmill1.gif 到 windmill6.gif。使用 JavaScript 使它们动起来。在 Student Disk 的 Tutorial.07 文件夹内将文档存为 Windmill.html。

4．使用 Paint 或其他图像程序创建三幅交通灯图像。一幅图像是绿灯，一幅图像是黄灯，一幅图像是红灯。使用 JavaScript 循环这三幅图。在 Student Disk 的 Tutorial.07 文件夹内将文档存为 TrafficLight.html。

5．在 Student Disk 的 Tutorial.07 文件夹内包含了两幅三角旗图像：pennant1.gif 和 pennant2.gif。使用 JavaScript 使它们动起来，让它们好像是在风中飘动。在 Student Disk 的 Tutorial.07 文件夹内将文档存为 Pennant.html。

6．使用 Internet 搜索引擎（如 Yahoo!）或其他能够使用的图像程序，为字母表中的每个字母找到一幅动物图像，如 A 是土豚，B 是海狸等。（为了搜索图像，请使用 clip art、images、pictures、.jpg 或.gif 关键字。确保所搜索的图像是在公共域内）使用 Paint 或其他图像程序为每幅图像添加字母表内的相应字母。在 Student Disk 的 Tutorial.07 文件夹内将

文档存为 AnimalAlphabet.html。

7.2 动画和层叠式表单

本节目标

在本节将学习：

- ✧ 层叠式表单
- ✧ 怎样在 Navigator 和 Internet Explorer 内使用 JavaScript 的样式
- ✧ 层叠式表单的定位
- ✧ 怎样在 Navigator 和 Internet Explorer 内使用定位
- ✧ 跨浏览器兼容性

7.2.1 层叠式表单

层叠式表单是用来创建 DHTML 的三项技术之一。层叠式表单（CSS，也称为样式表单）是 World Wide Web 协会用来管理 HTML 文档的格式化信息的标准集。与文档的内容相对，格式化信息包括字体、背景、颜色、布局和 HTML 文档的外观的其他特性。格式化信息的每个片断称为样式。在样式出现之前，页面设计人员将格式化应用于每个 HTML 标签来控制页面的外观。通过 CSS，页面设计人员能够为每个 HTML 页进行集合管理和创建一致的、跨越不同页面的外观。本质上，CSS 是将文档的内容和它的外观分隔开来。

DHTML 使用 JavaScript 来管理 CSS 和动态地改变标签的外观和 HTML 文档内的元素的位置。例如，页面具有一系列的按钮，用户能够单击它们来根据他们个人偏爱来改变页面的外观。在此节的后面部分，将学习怎样在页面上动态地重新定位 HTML 元素和使用 CSS 来创建行进的动画。

提示：网页设计人员最感兴趣的是 CSS，因为它比 HTML 能够对网页的可视的外观进行更多的控制和管理。学习 CSS 的目的是使用 JavaScript 和 DHTML 让网页更有动态性。因此，本节仅粗略地介绍一下 CSS 的样式和语法。想了解关于 CSS 的更多的内容，请访问 World Wide Web 协会的网址 http://www.w3.org。

使用由冒号分隔的名称/值对来创建 CSS 样式。名称/值对的名称部分用来引用特定的 CSS 样式属性（Property）。每个 CSS 属性能够利用不同的值（取决于属性）来格式化。表 7-4 包含了常用的 CSS 属性的描述和实例。

CSS 样式的两种类型是：内嵌样式和文档级样式表单。内嵌样式确定了 HTML 文档内单个标签的外观。使用 STYLE 属性和包含了想使用的样式的名称/值对的字符串来定义内

嵌样式。多个属性之间由分号分隔。下面是将<H1>标签的字体设置为 serif、字体宽度设置为粗体、字体样式设置为斜体的内嵌样式的代码。

```
<H1 STYLE="font-family: serif; font-weight: bold;
    font-style: italic>
```

表 7-4　一般的 CSS 属性

CSS 属 性	描　述	代 码 实 例
border-color	指定元素的边界颜色	border-color: blue;
font-family	指定字体的名称	font-family: arial;
font-style	指定字体样式：常规、斜体	font-style: italic;
font-weight	指定字体的宽度：常规、粗体、黑体、舒体或 100 和 900 之间的数字；400 与常规相同，700 与粗体相同	font-weight: bold;
margin-left	调整左页边空白	margin-left: 1in;
margin-right	调整右页边空白	margin-right: 1in;
text-align	决定文本的对齐方式：居中、分散对齐、左对齐或右对齐	text-align: center;

文档级样式表单决定了 HTML 标签的全局格式。在文档的<HEAD>部分嵌入<STYLE>…</STYLE>标签对集合来创建文档级表单。特定标签的任何格式命令都将被应用到包含在文档体内的所有相关的标签。样式表单应用内特定样式所控制的标签称为选择器。若每个选择器的规则都用一对{}括起来，那么能够在样式表单内包括多个选择器。与内嵌样式相同，选择器的多个属性之间由分号分隔。下面的代码显示了一个样式表单实例，它对<H1>、<H2>和<BODY>标签应用样式。每个标签后跟着一对包含了样式命令的括号。文档体内的所有标签实例都使用这些命令来格式化。

```
<STYLE>
H1 {color: red; font-family: "ms sans serif";
    font-size: 24pt; font-weight: bold;}
H2 {color: black; font-family: "ms sans serif";
    font-size: 18pt; margin-top: 0.25in; margin-left: 1in;}
BODY {color: blue; font-family: "arial";
    font-size: 12pt; font-weight: medium;}
</STYLE>
```

HTML 文档内的标签应用样式的另一种方法是使用 CLASS 属性。CLASS 属性能够被用于任何 HTML 标签，并确认各种元素是同一组的一部分。CLASS 属性的语法是<TAG CLASS="class name">。将语法的 TAG 部分替换为想作为类的一部分包括进去的 HTML 标签。为了将<H1>标签和<P>作为 myClass 类的一部分，使用语法<H1 CLASS="myClass">和

<P CLASS="myClass">。在 HTML 文档内能够创建两种类型的 CSS 类：常规类和通用类。常规类用来为相同的标签定义不同的样式命令。例如，假设在文档内需要两种或多种样式的<H1>标签。通过在样式后添加句点和类名在<STYLE>…</STYLE>标签对内创建常规类。接着在文档体内必要的标签内包括合适的类名。下面的代码包含了两个常规类：level1 和 level2，它们为<P>标签的不同实例定义了格式化命令。

```
<STYLE>
P.level1 {color: black; font-family: "serif";
        font-size: 10pt;font-weight: medium; text-indent: 1in}
P.level2 {color: green; font-family: "serif";
        font-size: 8pt;font-style: italic; text-indent: 2in}
</STYLE>
<BODY>
<P CLASS="level1">This is the level1 class</P>
<P CLASS="level2">This is the level2 class</P>
</BODY>
```

通用类与常规类类似，不同之处在于它不与任何的标签关联在一起。可以在<STYLE>…</STYLE>标签对内使用前面跟着句点的类名来创建通用类。下面的代码是名为 highlight 的通用类实例，它用黄色、arial 和蓝色背景来格式化文本。注意在体内部分，通用类被应用于两种样式：粗体<B>和加重<STRONG>。

```
<STYLE>
.highlight {color: yellow; font-family: arial;
        background: blue}
</STYLE>
<BODY>
<P>This <B CLASS="highlight">line</B> contains two
<STRONGCLASS="highlight">examples</STRONG>
of the generic class.</P>
</BODY>
```

ID 属性也可以被用于样式。ID 属性值惟一地标识了 HTML 文档内的单个标签。ID 属性的语法是<TAG ID="unique name">。ID 与类相似，不同之处在于它仅用于 HTML 文档体内的使用特定 ID 属性的某一处。在<STYLE>…</STYLE>标签对内创建 ID 属性，它的 ID 名前面是数字标号#，如下面的实例所示：

```
<STYLE>
#biggreenline {color: green; font-family: arial;
        font-size: 24pt}
</STYLE>
```

```
<BODY>
<P ID="biggreenline">The formatting for this
        ID class applies only to this line.</P>
</BODY>
```

为了说明使用样式管理 HTML 文档的外观是多么的容易，图 7-24 是不使用样式表单的 HTML 文档的一个实例。注意格式化<H1>和<H2>标签的语句在每次使用此标签时都被重复。

```
<HTML>
<HEAD>
<TITLE>Example of Individually Formatted Tags</TITLE>
</HEAD>
<BODY>
<H1><FONT SIZE=6 COLOR="red" FACE="arial">
First instance of Heading 1</FONT></H1>
<H2><FONT SIZE=5 COLOR="black" FACE="arial"><I>
First instance of Heading 2</FONT></I></H2>
<H1><FONT SIZE=6 COLOR="red" FACE="arial">
Second instance of Heading 1</FONT></H1>
<H2><FONT SIZE=5 COLOR="black" FACE="arial"><I>
Second instance of Heading 2</FONT></I></H2>
</BODY>
</HTML>
```

图 7-24　单个格式化的标签

现在查看图 7-25 中使用了 CSS 样式的相同的文档。注意样式表单是怎样将文档内容和它的外观分隔开的。若想改变元素的外观，如<H1>标签的颜色，那么仅需要在样式表单内修改一次，而不需要在文档内任何使用标签的地方都进行修改。在图 7-26 中显示了浏览器内文档的两个输出结果。

7.2.2　在 JavaScript 中使用 CSS 样式

CSS 样式决定了 HTML 文档元素的格式。为了在浏览器显示了文档之后修改 CSS 样式，可以使用 JavaScript。如前面的 7.1 节所讲的，不存在能够在 Internet Explorer 和 Navigator 中都能运行的兼容的 DHTML 标准。在使用 JavaScript 管理 CSS 样式时，这种不兼容性十分明显。Internet Explorer 和 Navigator 支持互不兼容的 Document 对象属性和方法。因为 JavaScript 使用 Document 对象属性和方法来访问 CSS 样式，若想使用 JavaScript 代码来管理 CSS，具有下面三个选择：

✧　编写仅能在 Navigator 中运行的代码。

```
<HTML>
<HEAD>
<TITLE>Example of Style Sheet Heading Tags</TITLE>
</HEAD>
<STYLE>
H1 {color: red; font-family: "ms sans serif";
      font-size: 24pt;}
H2   {color: black; font-family: "ms sans serif";
      font-size: 18pt; font-style: italic}
</STYLE>
<BODY>
<H1>First instance of Heading 1</H1>
<H2>First instance of Heading 2</H2>
<H1>Second instance of Heading 1</H1>
<H2>Second instance of Heading 2</H2>
</BODY>
</HTML>
```

图 7-25　样式表单的标题标签

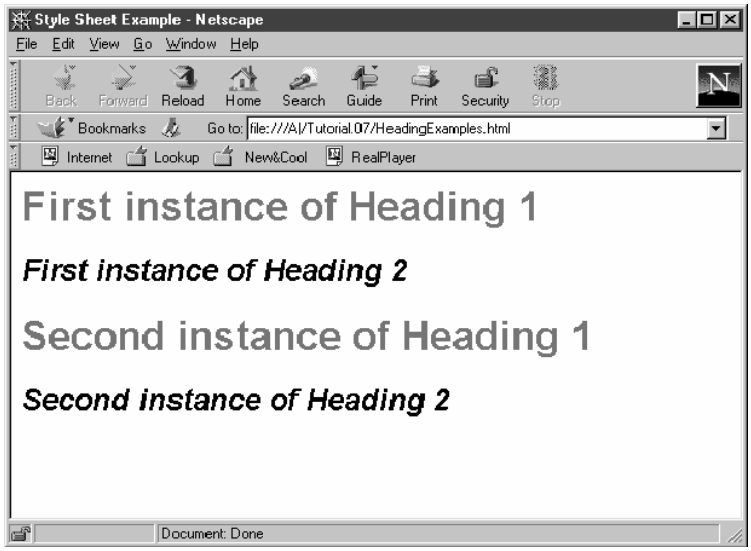


图 7-26　标题样式实例

✧　编写仅能在 Internet Explorer 中运行的代码。

✧　编写两种代码，根据使用的浏览器来执行正确的代码。

对每一种浏览器，本节仅简要地介绍了怎样使用 JavaScript 来引用 CSS 样式和编写在两种浏览器中都能运行 DHTML 代码的技术。因为 DHTML 发展迅速，不必为了在 JavaScript 中使用 CSS 而在 Microsoft 和 Netscape 特有的方法上花费太多的时间。记住当两种浏览器完全支持 DHTML 标准的 1 级时，现在使用的让 JavaScript 在 Microsoft 和 Netscape 内都能

与 CSS 交互的方法将彻底地改变或变得过时。

在 Navigator 中使用 JavaScript 和 CSS 样式

Navigator 文档对象模型使用 Document 对象的标签、类和 ID 属性来访问样式。tags 属性能够访问 HTML 文档内的标签样式。classes 属性能够访问 HTML 文档内的类样式。ids 属性能够访问 HTML 文档内的 ID 属性样式。

提示：tags、classes 和 ids 属性仅能够在 Navigator 中使用。

为了在 Navigator 中引用 CSS 样式，需要在 Document 对象后添加句点和 tags、classes 和 ids 属性，接着是句点和 CSS 选择器的名称，再接着是另一个句点和 CSS 属性。在 Navigator 中引用 CSS 样式的通用的语法是 document.tags（classes 或 ids 属性）.selector.属性。当在 JavaScript 代码中引用包含连字符的 CSS 属性时， 需要删除连字符并将首词转换为小写形式，将后续词的首字母转换为大写形式。以全部为小写字母的形式来引用不具有连字符的 CSS 属性。例如，用 border 来引用 border，用 borderColor 引用 border-color，用 fontSize 引用 font-size。

下面的实例使用 JavaScript 语法来为 tags、classes 和 ids 属性来定义样式。此代码执行的功能与位于<STYLE>…</STYLE>标签对之间的标准的 CSS 语法相同。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
document.tags.H1.color = "red";
document.tags.H1.fontSize = "24pt";
document.classes.level1.color = "black";
document.classes.level1.fontFamily = "serif";
document.ids.biggreenline.color = "green";
document.ids.biggreenline.fontFamily = "arial";
document.ids.biggreenline.fontSize = "24pt";
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
```

尽管 Navigator 允许使用 JavaScript 来定义样式，但是不能动态地改变样式值。能够在浏览器显示它之后使用 JavaScript 代码来改变样式的值，但是直到用户改变窗口的大小之前，改变将不会被显示。

在 Internet Explorer 中使用 JavaScript 和样式

Internet Explorer 文档对象模型使用 Document 对象的 all 属性来访问样式。all 属性是包含了 HTML 文档内的所有元素的数组。将前有句点的 all 属性添加到 Document 对象后，接着跟着句点和特定的 CSS 样式的名称。接着添加句点和 style 属性，紧跟着是句点和特定的 CSS 属性。style 属性代表 CSS 样式的特殊标签、类或 ID。在 Internet Explorer 中引用 CSS 样式的通用的语法是 document.all.selector.style.属性。

提示：all 和 style 属性仅能在 Internet Explorer 中使用。

和在 Navigator 中一样，当在 Internet Explorer 中使用 JavaScript 代码引用 CSS 属性时，需要从属性名中删除连字符并将属性转换为大小写混合的形式。下面的代码演示了在 Internet Explorer 中引用 CSS 样式的 JavaScript 语法。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
document.all.H1.style.color = "red";
document.all.H1.style.fontSize = "24pt";
document.all.level1.style.color = "black";
document.all.level1.style.fontFamily = "serif";
document.all.biggreenline.style.color = "green";
document.all.biggreenline.style.fontFamily = "arial";
document.all.biggreenline.style.fontSize = "24pt";
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
```

在 Internet Explorer 中，必须使用 JavaScript 和 Document 对象的 styleSheets[]数组来定义新的 CSS 样式。styleSheets[]数组使用起来有点麻烦，因此建议在<STYLE>…</STYLE>标签对之间使用 CSS 语法来定义样式。然而，在 Internet Explorer 中能够使用 JavaScript 来动态改变样式。当在 Internet Explorer 中使用 JavaScript 改变样式时，变化将立即在浏览器中显示出来，而不像在 Navigator 中那样，必须改变窗口的大小。

7.2.3 CSS 定位

在 7.1 节中，使用<IMG>标签和 JavaScript 创建简单的动画。但是<IMG>标签十分有限，因为仅能够运行位置固定的动画，即使用<IMG>标签创建的动画不能在屏幕上行进。实际上，若不从服务器上重新下载 HTML 文档，那么将不能在 Web 页面上重新定位图像。CSS 技术推荐的扩展称为 CSS 定位。CSS 定位用于定位或在 Web 页面上对元素进行布局。尽管是 CSS 推荐的扩展，但是在 Navigator 和 Internet Explorer 中已经支持 CSS 定位。因为在两种浏览器之间 Document 对象的不兼容性，所以不能编写能够在两种浏览器中都能运行的 JavaScript 代码。必须为某种浏览器或者是另外一种浏览器编写程序，或编写两套代码并且根据使用的浏览器的类型来决定运行哪套代码。

有两种类型的 CSS 定位：相对和绝对。相对定位根据 Web 页面中的其他元素来放置元素。绝对定位将元素放置在 Web 页面中的特定的位置内。相对定位主要用于 Web 页面的设置和布局，它不在本教程介绍的范围内。使用 JavaScript 和绝对定位的目的之一是创建完整动画。

通常将定位应用于具有内嵌样式的标签。还能够在文档级样式表单中使用 CSS 定位。

然而，若将定位用于文档级样式表单内的标签（如<H1>），那么接着文档中的<H1>标签的所有实例都将定位在相同的位置上。相比之下，为文档级样式表单的 ID 使用定位也能正常运行。然而以内嵌样式直接定位元素通常更容易些。

表 7-5 列出了几个常用的 CSS 定位属性。

表 7-5　常用的 CSS 定位属性

属　　性	描　　述	值
position	决定元素被定位的方式	绝对或相对
left	从窗口的左上角起的水平距离	像素数
top	从窗口的左上角起的垂直距离	像素数
width	边界框的宽度	像素数
height	边界框的高度	像素数
visibility	决定元素是否可见	可见或隐藏

Navigator 不能为非容器元素识别 CSS 定位——即不具有结束标志的元素。例如，对<IMG>标志使用 CSS 定位，因为它不具有结束标志。为了维护与 Internet Explorer 的兼容性，被定位的元素通常被放置在<SPAN>…</SPAN>或<DIV>…</DIV>标签对内。<SPAN>标签用于为 HTML 文档段应用格式，而<DIV>标签将文档分为不同的段。特定的 CSS 属性被放置在<SPAN>或<DIV>起始标签的尖括弧内。

下面的代码以绝对方式定位包含在<SPAN>标签对内的图像。浏览器内显示的输出图像如图 7-27 所示。

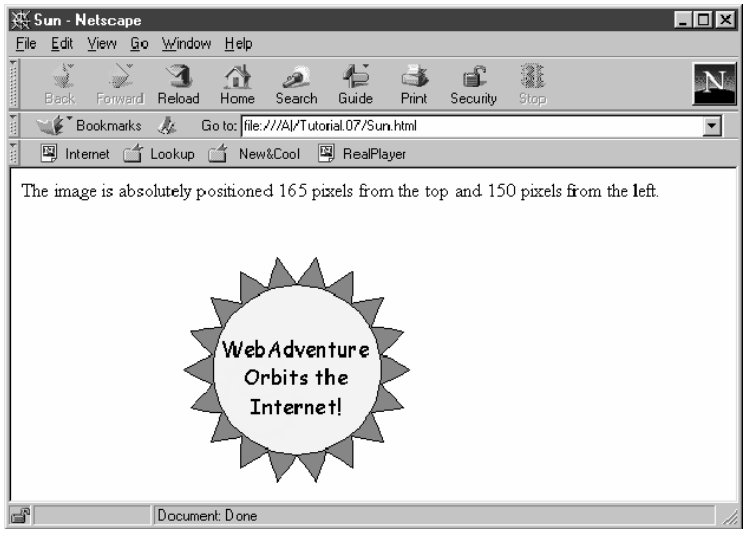


图 7-27　CSS 定位

```
<SPAN STYLE="position:absolute; left:150; top:165">
<IMG SRC="sun.gif">
```

```
</SPAN>
```

下一步，将在 HTML 文档内定位两幅图像。图像（名为 up.gif 和 down.gif）位于 Student Disk 的 Tutorial.07 文件夹内。up.gif 文件是展翅的飞鸟图像，而 down.gif 文件是收翅的飞鸟图像。所创建的 Web 页中使用了 up.gif 图像两次而使用 down.gif 图像一次。将定位三幅图像让鸟看起来像是在飞翔。

在 HTML 文档内定位两幅图像：

1．启动文本编辑器或 HTML 编辑器，并创建新文档。

2．输入文档的<HTML>和<HEAD>段。

```
<HTML>
<HEAD>
<TITLE>Two Birds</TITLE>
</HEAD>
```

3．输入<BODY>编写文档段。

4．添加下列行创建第一幅绝对定位的图像。图像 up.gif 被包含在<SPAN>…</SPAN>标签对内。使用 STYLE 属性来设置图像的绝对位置：从左开始 40 个像素，从顶端开始 200 个像素。

```
<SPAN STYLE="position:absolute;left:40;top:200">
<IMG SRC="up.gif">
</SPAN>
```

5．添加下列行创建第二幅绝对定位的图像。图像 down.gif 也被包含在<SPAN>…</SPAN>标签对内。使用 STYLE 属性来设置图像的绝对位置：从左开始 250 个像素，从顶端开始 80 个像素。

```
<SPAN STYLE="position:absolute;left:250;top:80">
<IMG SRC="down.gif">
</SPAN>
```

6．现在添加下列行创建第三幅绝对定位的图像。再次使用了 up.gif 图像，它被定位在从左开始 480 个像素，从顶端开始 10 个像素的位置上。

```
<SPAN STYLE="position:absolute;left:480;top:10">
<IMG SRC="up.gif">
</SPAN>
```

7．添加下面的代码结束<BODY>和<HTML>标签。

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

8．在 Student Disk 的 Tutorial.07 文件夹内将文件保存为 TwoBirds.html。在浏览器中打开 TwoBirds.html 文件。图 7-28 为输出的结果。

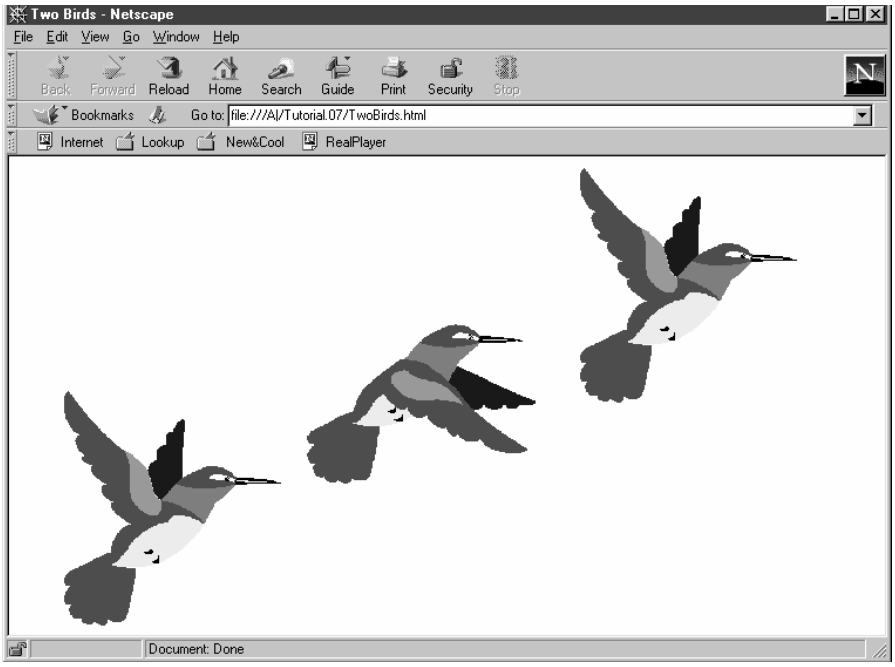


图 7-28　TwoBirds.html 的输出

9．关闭 Web 浏览器窗口。

下面，将学习怎样使用 JavaScript 动态定位图像来创建行进的动画。因为 Navigator 和 Internet Explorer 之间的不兼容性，所以将学习怎样为每种浏览器实现动态定位。首先，学习在 Internet Explorer 中动态定位。

7.2.4　在 Internet Explorer 中定位

如前面所学的，Internet Explorer 让您能够使用 JavaScript 来动态改变 CSS 的样式。文档外观的改变将立即被显示。既然 CSS 定位是 CSS 的扩展，所以能够使用 CSS 的 left 和 right 属性来动态改变元素在屏幕上的位置。例如，语句 document.all.sampleimage.style.left = "3.00in";通过将 left 属性的值改变为“ 3.00in ”将 ID 为 sampleimage 的元素移动到距左边 3 英寸的地方。将 CSS 的 left 和 right 属性与 setTimeout()或 setInterval()方法结合起来，能够创建行进的动画。

作为 Internet Explorer 中行进动画的 CSS 实例，需要查看在前面的练习中鸟的简单的

动画。程序使用 setInterval()方法和绝对定位使两幅飞鸟图像运动起来。因为程序仅包含了三个不同位置的两幅图像，所以它并不令人激动。然而，它让您理解了怎样创建行进的动画。

鸟的动画程序显示在图 7-29 中。创建了两个 Image 对象（up 和 down）来保存每幅图像文件。用户单击 Fly 按钮来启动行进的动画。Fly 按钮的事件处理器 onClick 事件调用了 setInterval()方法，它执行 fly()函数。fly()函数使用了三个 if 语句来执行动画。每条 if 语句判断 position 变量的值。若 position 属性等于 1 或 3，那么 birdimage <IMG>标签的 SRC 属性将通过 document.birdimage.src = up.src;语句被改变为 up.gif。若 position 属性等于 2，那么 birdimage <IMG>标签的 src 属性将通过 document.birdimage.src = down.src;语句被改变为 down.gif。每条 if 语句还改变了 bird（包含了 down.gif 图像）<SPAN>标签的绝对位置的 left 和 top 属性。position 变量接着被设置为 2 或 3。在第三个位置显示之后，position 属性重置为 1，接着再启动动画。

提示：若想查看飞翔的飞鸟动画在 Internet Explorer 中是怎样运行的，在 Student Disk 的 Tutorial.07 目录内包含了 FlyingBirds.html 程序。请记住只能在 Internet Explorer 中打开此文件。若在 Navigator 中打开，将会弹出错误消息。

下一步，为了说明实现行进的动画是多么地简单，将创建用于 Internet Explorer 的地球绕太阳旋转的动画。地球旋转的轨道的坐标按图 7-30 进行划分。注意绕太阳的位置并不十分精确。为了计算精确的位置，需要一个复杂的公式计算轨道的半径。还请注意在地球的轨道上使用更多的位置能够使动画更平滑。

提示：即使未安装 Internet Explorer，也需要创建程序，因为在学习跨越浏览器兼容性时，将在本节的后面部分使用此程序。

创建 Internet Explorer 下的轨道动画：

1．启动文本编辑器或 HTML 编辑器，并创建新文档。

2．输入文档的<HTML>和<HEAD>段。

```
<HTML>
<HEAD>
<TITLE>Orbit</TITLE>
</HEAD>
```

3．为 JavaScript 段添加起始语句。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

4．创建保存 16 个轨道位置中的当前位置的变量，添加语句 position = 0;。

```
<HTML>
<HEAD>
<TITLE>Flying Bird</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
var startFlying;
var updown = 0;
var horizontalPosition = 10;
var up = new Image();
up.src = "up.gif";
var down = new Image();
down.src = "down.gif";
position = 1;
function fly() {
        if (position == 1) {
                document.birdimage.src = up.src;
                document.all.bird.style.left = 40;
                document.all.bird.style.top = 200;
                position = 2;
        }
        else if (position == 2) {
                document.birdimage.src = down.src;
                document.all.bird.style.left = 250;
                document.all.bird.style.top = 80;
                position = 3;
        }
        else if (position == 3) {
                document.birdimage.src = up.src;
                document.all.bird.style.left = 480;
                document.all.bird.style.top = 10;
                position = 1;
        }
}
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
<BODY>
<SPAN ID="bird" STYLE="position:absolute; left:40; top:200">
<IMG NAME="birdimage" SRC="up.gif" HEIGHT=200 WIDTH=200>
</SPAN>
<FORM>
<INPUT TYPE="button" NAME="fly" VALUE=" Fly "
onClick="startFlying=setInterval('fly()',500);">
<INPUT TYPE="button" NAME="stop" VALUE=" Stop "
onClick="clearInterval(startFlying);">
</FORM>
</BODY>
</HTML>
```

图 7-29　飞鸟动画文档

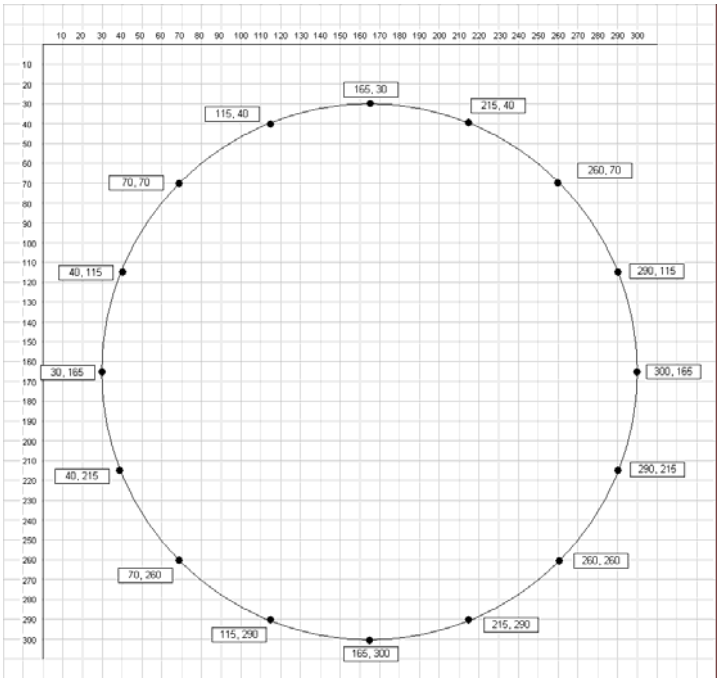


图 7-30　Orbit 动画坐标

5．创建下面的数组来保存与图像的 left 位置对应的 16 个位置。此位置与图 7-30 中划分的每对位置集中的第一个位置相对应。

```
leftEarth =new Array(16);
leftEarth[0] = 165; leftEarth[1] = 215; leftEarth[2] = 260;
leftEarth[3] = 290; leftEarth[4] = 300; leftEarth[5] = 290;
leftEarth[6] = 260; leftEarth[7] = 215; leftEarth[8] = 165;
leftEarth[9] = 115; leftEarth[10] = 70; leftEarth[11] = 40;
leftEarth[12] = 30; leftEarth[13] = 40; leftEarth[14] = 70;
leftEarth[15] = 115;
```

6．创建下面的数组来保存与图像的 top 位置对应的 16 个位置。此位置与图 7-30 中划分的每对位置集中的第二个位置相对应。

```
topEarth =new Array(16);
topEarth[0] = 30; topEarth[1] = 40; topEarth[2] = 70;
topEarth[3] = 115; topEarth[4] = 165; topEarth[5] = 215;
topEarth[6] = 260; topEarth[7] = 290; topEarth[8] = 300;
topEarth[9] = 290;
topEarth[10] = 260; topEarth[11] = 215; topEarth[12] = 165;
topEarth[13] = 115; topEarth[14] = 70;
```

```
topEarth[15] = 40;
```

7．添加下面的 orbit()函数，它是在 leftEarth 和 topEarth 数组的 16 个位置之间循环地球图像。由 position 变量来跟踪每个位置。在<BODY>标签的 onLoad 事件处理器中使用 setInterval()方法来调用 orbit()函数。

```
function orbit(){
        document.all.earth.style.left = leftEarth [position ];
        document.all.earth.style.top = topEarth [position ];
        ++position;
        if (position == 16)
                position = 0;
}
```

8．添加下面的代码结束<SCRIPT>和<HEAD>段。

```
//STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
```

9．添加下面的<BODY>标签的起始代码，它使用 setInterval()方法在 onLoad 事件处理器内执行 orbit()函数。

```
<BODY onLoad="setInterval('orbit()',200)">
```

10．输入下面的<SPAN>段，它保存了太阳的位置和并对它进行定位。

```
<SPAN STYLE="position:absolute;left:95;top:95">
<IMG SRC="sun.gif">
</SPAN>
```

11．添加另一个<SPAN>段来保存和初始化地球图像的位置。因为是使用 CSS 定位让地球运动起来，所以在<SPAN>段给出了值为 earth 的 ID。

```
<SPAN ID="earth"STYLE="position:absolute;left:165;
        top:30">
<IMG SRC="earth.gif">
</SPAN>
```

12．输入</BODY>和</HEAD>结束标签。

13．在 Student Disk 的 Tutorial.07 文件夹内将文件存为 OribitNavigator.html。若安装

Internet Explorer，那么打开文件，观察动画运行是否正常。图 7-31 显示了在动画开始前程序在 Navigator 中的输出结果。

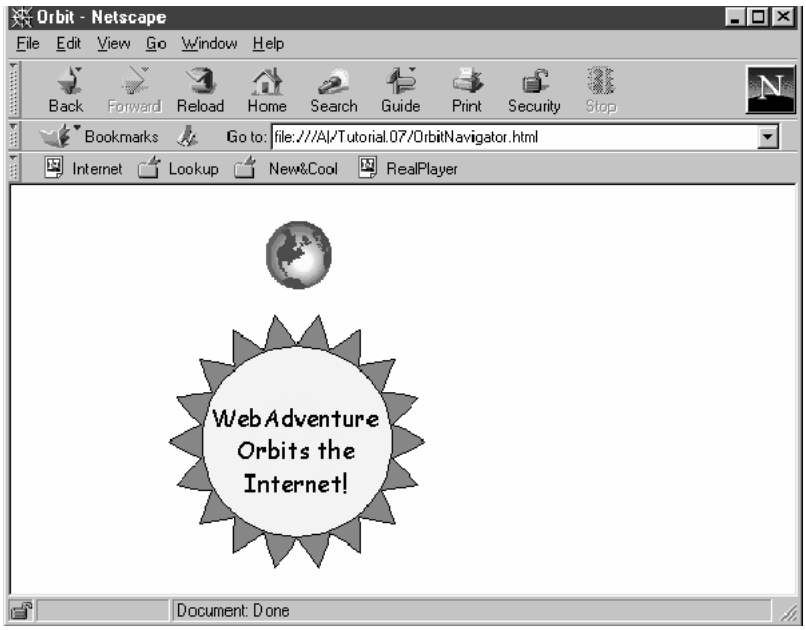


图 7-31　在 Navigator 中的 OrbitNavigator.html 文档

14．关闭浏览器窗口。

7.2.5　在 Navigator 中定位

Navigator 不会动态动画使用 CSS 定位。相反，必须使用 layers。在 Navigator 中使用 layers 在段中排列 HTML 元素，它们能够放置在其他段的上面，也能够单个地移动。仍能够在 Navigator 中使用 CSS 定位，但是不能用于行进的动画。尽管分层不是 CSS 协议的一部分，但是需要理解 layers 能够被用来在 Navigator 中创建行进的动画。分层是一个大话题，它不仅能够被用于动画，而且也能够被用来解决 Navigator 中的其他 HTML 设计问题。然而，将仅讨论分层方面与动画相关的的内容。

能够使用<LAYER>…</LAYER>标签对在 HTML 文档内创建一层。使用<LAYER>标签的 LEFT 和 TOP 属性来指定层的初始位置。还能够在<LAYER>标签内包括 NAME 属性。

JavaScript 使用 Layer 对象来访问每个<LAYER>标签。JavaScript 内 Layer 对象具有几个属性和维护层的方法。在 Navigator 中创建行进动画的 Layer 对象的两个方法是 moveTo()方法和 offset()方法。moveTo()方法将层移至指定的位置，它接受两个参数。第一个参数代表从窗口的左边开始的像素数，第二个参数代表从窗口的顶端开始的像素数。offset()方法将层从当前位置在水平方向和垂直方向移动指定的像素数。offset()方法也接受两个参数。第一个参数代表水平移动的像素数，第二个参数代表垂直移动的像素数。

使用 layers[]数组中的位置或使用赋给<LAYER>标签的 NAME 属性值来在 JavaScript 中引用特定的层。与 JavaScript 中其他的数组一样，如 forms[]数组，层按照 JavaScript 解释器解释的顺序被赋给 layers[]数组。为了引用文档中的第一个层，使用语句 document.layers[0];。然而，使用赋给它的 NAME 属性值来引用它更容易些。例如，要引用 animation 数组，使用语句 document.animation;。每个层都包含它自己的 Document 对象，必须包含它来引用层的元素，因此使用了 Document 对象两次。例如，要改变 animation 层内 myImage 图像的 src 属性，使用语句 document.animation.document.myImage.src = "new_image.jpg";。

图 7-32 显示了在图 7-29 中所看到的相同的飞鸟动画文档，但是为了在 Navigator 中使用它对它进行了修改。在 fly()函数中使用语句 document.bird.document.birdimage.src = up.src; 和 document.bird.document.birdimage.src = down.src;来改变显示的图像。在每个 if 语句内的每条语句使用 moveTo()方法来移动 bird 层，它包含了飞鸟图像。

提示：想观察飞翔的鸟动画函数在 Navigator 中是怎样运行的，在 Student Disk 的 Tutorial.07 文件夹内包含了 FlyingBirdNavigator.html 文件。请记住只能在 Navigator 中打开此文件。若是在 Internet Explorer 中打开此文件，将弹出错误消息。

下一步，修改轨道程序让它能在 Navigator 下运行。与在 Internet Explorer 中一样，即使未安装 Navigator，仍需要创建程序，因为在学习跨浏览器兼容性时，将需要它。

修改轨道动画程序让它能在 Navigator 下运行：

1．在文本编辑器或 HTML 编辑器中打开 OrbitIE.html 文件，并在 Student Disk 的 Tutorial.07 文件夹内将它另存为 OrbitNavigator.html。

2．在 orbit()函数内，为 Earth 图像用下面的单条语句替换修改<SPAN>标签的 left 和 top 属性的两条语句。新语句使用 Navigator 的 moveTo()方法来改变 earth 层的位置，添加下列文本：

```
document.earth.moveTo(leftEarth[position],
        topEarth[position]);
```

3．使用下面的<LAYER>标签替换用于 earth 图像的<SPAN>段。

```
<LAYER NAME="earth"LEFT=165 TOP=30>
<IMG SRC="earth.gif">
</LAYER>
```

4．保存文档。若安装了 Navigator，打开此文档并观察它运行是否正常。图 7-33 显示了在动画开始之前程序在 Navigator 中的输出结果。

```
<HTML>
<HEAD>
<TITLE>Flying Bird</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
var startFlying;
var updown = 0;
var horizontalPosition = 10;
var up = new Image();
up.src = "up.gif";
var down = new Image();
down.src = "down.gif";
position = 1;
function fly() {
        if (position == 1) {
                document.bird.document.birdimage.src = up.src;
                document.bird.moveTo(40, 200);
                position = 2;
        }
        else if (position == 2) {
                document.bird.document.birdimage.src = down.src;
                document.bird.moveTo(250, 80);
                position = 3;
        }
        else if (position == 3) {
                document.bird.document.birdimage.src = up.src;
                document.bird.moveTo(480, 10);
                position = 1;
        }
}
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
<BODY>
<LAYER NAME="bird" LEFT=40 TOP=200>
<IMG NAME="birdimage" SRC="up.gif" HEIGHT=200 WIDTH=200>
</LAYER>
<FORM>
<INPUT TYPE="button" NAME="fly" VALUE=" Fly "
        onClick="startFlying=setInterval('fly()',500);">
<INPUT TYPE="button" NAME="stop" VALUE=" Stop "
        onClick="clearInterval(startFlying);">
</FORM>
</BODY>
</HTML>
```

图 7-32　在 Navigator 中使用的修改过的飞鸟动画文档

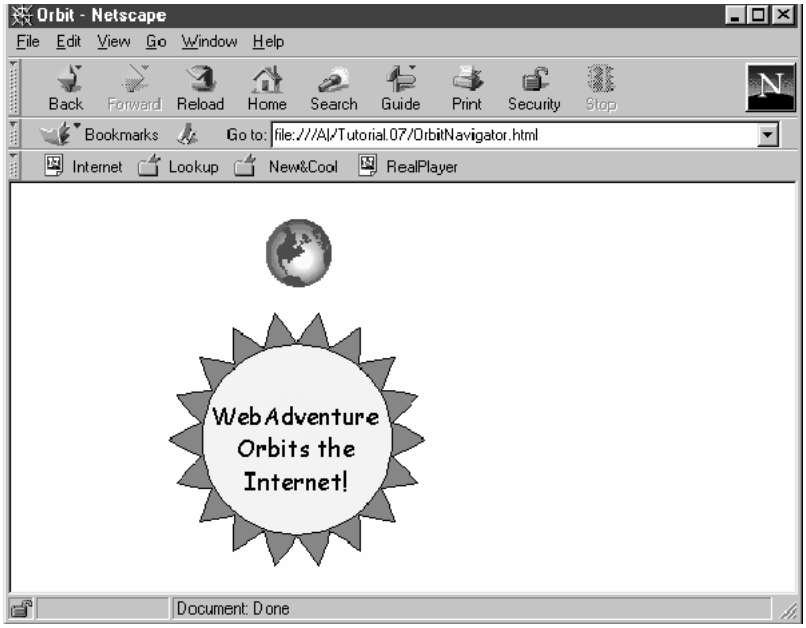


图 7-33　Navigator 中的 OrbitNavigator.html

5．关闭 Web 浏览器窗口。

7.2.6　跨浏览器兼容性

人们和企业都想让他们的 Web 页能够吸引访问者，都想让 Web 页尽可能地吸引人和十分有趣味。约 46%的 Internet 用户使用 Navigator，约 42%的用户使用 Internet Explorer，其余约 12%的用户使用其他的浏览器。若开发人员不得不要选择某种浏览器，那么大部分 Internet 用户将不能浏览他们的网址。许多开发人员试图创建既能在 Navigator 也能在 Internet Explorer 中正常运行的 DHTML 代码。然而，创建真正的跨浏览器兼容的 DHTML 文件是一项艰巨的任务，它需要深入理解 Microsoft 和 Netscape 的文档对象模型。另外，必须编写能在用户浏览器下运行的语句的代码。

一种比试图将与 Microsoft 和 Netscape 都兼容的代码封装在相同的文档内更容易的方法是创建两个独立的文档，每种浏览器对应一个文档。接着能够使用“ 主 ”文档来检查用户打开文件时正在运行哪种浏览器。在“ 主 ”文档得知正在使用的浏览器之后，它将打开合适的 Web 页。使用在第 5 章学习的 Navigator 对象的 appName 属性来判断正在运行的浏览器。若 appName 属性返回‘ Netscape ”,“ 主 ”文档打开在 Netscape 下运行的文件。若 appName 属性返回“ Internet Explorer ”，“ 主 ”文档打开在 Internet Explorer 下运行的文件。

图 7-34 中的程序打开在特定浏览器下运行的飞翔的飞鸟动画程序。<BODY>标签内的 onLoad()事件处理器调用 checkBrowser()函数。checkBrowser()函数使用 Navigator 对象的 appName 属性来判断正在运行哪种浏览器。接着使用 document.location.href;语句来打开正确的文件。

```
<HTML>
<HEAD>
<TITLE>Flying Bird</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
function checkBrowser() {
        if (navigator.appName == "Netscape")
                document.location.href = "FlyingBirdNavigator.html";
        else
                document.location.href = "FlyingBirdIE.html";
}
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
<BODY onLoad="checkBrowser()">
</BODY>
</HTML>
```

图 7-34　在检查 appName 之后打开特定浏览器文件的程序

许多 JavaScript 程序员不去检查正在使用的浏览器，而宁愿测试哪种类型的 DOM 正在使用。检测 Document 对象是否具有 layers 属性或 all 属性来判断正在使用哪种类型的 DOM。能够使用条件语句如 if(document.layers)和 if(document.all)来检测 layers 和 all 属性。例如，因为仅有 Navigator Document 对象包括了 layers 对象，所以若在 Navigator 下运行，语句 if(document.layers != null)（判断对象是否等于 null）将返回真；若在 Internet Explorer 下运行语句 if(document.layers != null)将返回假。同样，因为仅有 Internet Explorer Document 对象包括 all 对象，所以若在 Internet Explorer 下运行，语句 if(document.all != null)将返回真；若在 Navigator 下运行语句 if(document.all != null)将返回为假。

下面的代码对图 7-34 中的 checkBrowser()函数进行了修改，用来判断 all 或 layers 对象。

```
function checkBrowser() {
        if (document.layers != null)
                document.location.href = "FlyingBirdNavigator.html";
        else if (document.all != null)
                document.location.href = "FlyingBirdIE.html";
}
```

提示：想要更好地了解跨浏览器 JavaScript 技术，请参阅 Dynamic Drive http://www.dynamicdrive.com 和 Dynamic Duo http://www.dansteinman.com/dynduo/。

下一步，将创建用来打开特定浏览器的轨道程序的主文档，它取决于所使用的浏览器。

文档检测 all 和 layers 对象而不是 Navigator 对象的 appName 属性。

创建用来打开特定浏览器的轨道程序的主文档，它取决于所使用的浏览器：

1．启动文本编辑器或 HTML 编辑器，并创建新文档。

2．输入文档的<HTML>、<HEAD>、<TITLE>段的起始语句。

```
<HTML>
<HEAD>
<TITLE>Orbit</TITLE>
```

3．在<HEAD>段内为 JavaScript 添加起始语句。

```
<<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

4．创建下面的函数来检测正在使用哪种浏览器。将在<BODY>标签内的 onLoad 事件内调用此函数。

```
function checkBrowser(){
        if (document.layers != null)
                document.location.href = "OrbitNavigator.html";
        else if (document.all != null)
                document.location.href = "OrbitIE.html";
}
```

5．添加下面的代码结束<SCRIPT>和<HEAD>段。

```
//STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
```

6．输入包括了调用 checkBrowser()函数的 onLoad 事件的<BODY>标签。

```
<BODY onLoad="checkBrowser()">
```

7．添加</BODY>和</HTML>结束标签。

8．在 Student Disk 的 Tutorial.07 文件夹内将文件存为 OrbitMaster.html。从 Navigator 或是 Internet Explorer 内打开 OrbitMaster.html 文件，判断程序是否打开了正确的文件。打开的文档名显示在 Internet Explorer 的 Address 框中或 Navigator 的 Location 框中。

9．关闭 Web 浏览器窗口。

7.2.7 总结

- ✧ 层叠式表单（CSS，也称为样式表单）是 World Wide Web 协会提出的标准，它们用来格式化 HTML 文档的内容。
- ✧ 每个格式化信息片断称为样式。
- ✧ 使用由冒号分隔的名称/值对来创建 CSS 样式。名称/值对的名称部分用来引用特定的 CSS 样式属性，称为属性。
- ✧ 内嵌样式决定了 HTML 文档内的单个标签的外观。使用 STYLE 属性和包含了想使用的样式的名称/值对的字符串来定义内嵌样式。多个属性之间由分号分隔。
- ✧ 文档级样式表单为特定的 HTML 元素（如<H1>标签）的实例、特定的 CLASS 和 ID 属性决定了格式。
- ✧ CLASS 属性能够被用于任何 HTML 标签，并确认各种元素是同一组的一部分。
- ✧ ID 属性值用来惟一地标识 HTML 文档内的单个标签。
- ✧ 在样式表单内特定样式所控制的标签、类或 ID 被称为选择器。
- ✧ 常规类用来为相同标签定义不同的样式指令。
- ✧ 通用类类似于常规类，但是它不与任何特殊的标签关联在一起。
- ✧ 使用 JavaScript 代码维护 CSS 有三种选择：编写仅能在 Navigator 下运行的代码，编写仅能在 Internet Explorer 下运行的代码，或编写两套代码，根据所使用的浏览器来执行正确的代码。
- ✧ Netscape 文档对象模型使用 Document 对象的 tags、classes 和 ids 属性来为选择器访问样式。tags 属性用来访问 HTML 文档内的标签样式。classes 属性用来访问 HTML 文档内的类样式。ids 属性用来访问 HTML 文档内的 IDS 属性样式。
- ✧ 当在 JavaScript 代码内引用 CSS 属性时，删除连字符，第一个词的首字符使用小写形式，后续词的首字符使用大写形式。
- ✧ 在 Navigator 内定义 CSS 样式的通用语法是 document.tags（classes 或 ids 属性）.CSS 选择器.特殊的 CSS 属性。
- ✧ Internet Explorer 文档对象模型使用 Document 对象的 all 属性来为选择器访问样式。all 属性是包含了 HTML 文档内所有元素的数组。style 属性代表特殊标签、类或 ID 属性的 CSS 样式。
- ✧ 在 Internet Explorer 中定义 CSS 样式的通用语法是 document.all.CSS 选择器.style.特殊的 CSS 属性。
- ✧ 用 CSS 定位来定位或布局 Web 页内的元素。
- ✧ 相对定位根据 Web 页内的其他元素来放置某个元素。
- ✧ 绝对定位在 Web 页内某个特定位置来放置元素。
- ✧ 在 Navigator 下使用 layers 来排列段内的 HTML 元素，它们能够被放置在其他段的上面并且能够被单独移动。

- ✧ JavaScript 内 Layer 对象具有数个属性和维护层的方法。
- ✧ moveTo()方法将层移至特定的位置。
- ✧ offset()方法将层从当前位置在水平或垂直方向上移动指定的像素数。
- ✧ 替代在同一文档内编写 Microsoft 和 Netscape 的代码的方法是创建两个独立的文档每种浏览器对应一个文档。接着编写判断正在运行哪种浏览器和打开相应的文件的代码。
- ✧ 许多 JavaScript 程序员不去检查正在使用的浏览器，而宁愿测试哪种类型的 DOM 正在使用。通过检测 Document 对象是否具有 layers 属性或 all 属性来判断正在使用哪种 DOM。

7.2.8　问题

1．用来引用特定的 CSS 样式的名称/值对的名称部分称为____。

a．属性

b．键

c．定义

d．样式模块

2．____样式决定 HTML 文档内的单个标签的外观。

a．标签

b．行

c．内嵌

d．实例

3．____样式表单决定了特定 HTML 元素（如<H1>标签）、特定的 CLASS 和 ID 属性的格式。

a．全局

b．文档级

c．类

d．内嵌

4．CLASS 属性____。

a．包含了一组 JavaScript 函数

b．管理给定的<SCRIPT>…</SCRIPT>内的所有数组

c．决定浏览器是 Internet Explorer 还是 Navigator

d．确认各种元素是同一组的一部分

5．ID 属性值____。

a．包含了用来改变背景颜色的颜色标识符

b．惟一地标识 HTML 文档内的单个标签

c．是 JavaScript 代码内所使用的标签类型

d．仅与 HTML 文档内的图像一起使用

6．样式表单应用内的特定样式所控制的的标签、类或 ID 被称为____。

a．标签

b．标记

c．指示器

d．选择器

7．____类被用来为相同的标签定义不同的样式命令。

a．常规

b．标准

c．样式

d．JavaScript

8．不与任何特殊的样式关联在一起的类被称为____类。

a．非常规

b．通用

c．独立

d．浮动

9．当改变样式表单内所包含的样式格式时，将发生什么事情？

a．什么也不发生。必须重新从服务器下载 HTML 文档以便变化产生作用

b．文档内所有与样式关联在一起的标签将自动被更新

c．仅与样式关联在一起的第一个标签实例将被自动更新

d．必须在每个与样式关联在一起的标签上按鼠标右键，接着从快捷菜单中选择更新

10．下面用来管理 CSS 的 JavaScript 程序中的哪条语句是正确的？

a．管理 CSS 的 JavaScript 程序在 Navigator 和 Internet Explorer 下都能正常运行

b．为 Internet Explorer 编写的管理 CSS 的 JavaScript 程序能够在 Navigator 下运行，但是为 Navigator 编写的管理 CSS 的 JavaScript 程序不能在 Internet Explorer 下运行

c．为 Navigator 编写的管理 CSS 的 JavaScript 程序能够在 Internet Explorer 下运行，但是为 Internet Explorer 编写的管理 CSS 的 JavaScript 程序不能在 Navigator 下运行

d．若想管理 CSS 样式，必须为每种浏览器编写不同的 JavaScript 代码

11．在 JavaScript 中怎样编写 font-weight？

a．FONT-WEIGHT

b．fontweight

c．FontWeight

d．fontWeight

12．使用 JavaScript 代码改变样式值，那么何时 Navigator 显示此变化？

a．立即

b．在 HTML 文档从服务器重新被下载之后

c．在用户改变窗口大小之后

d．在 Navigator 下不能使用 JavaScript 来改变样式值

13．怎样在 Internet Explorer 下使用 JavaScript 代码来改变<H1>标签的颜色样式？

a．document.H1.style.color = "red"

b．document.all.H1.style.color ="red"

c．document.all.H1.color = "red"

d．document.H1.color = "red"

14．使用 JavaScript 代码改变样式值，那么何时 Internet Explorer 显示此变化？

a．立即

b．在 HTML 文档从服务器重新被下载之后

c．在用户改变窗口大小之后

d．在 Internet Explorer 下不能使用 JavaScript 来改变样式值

15．____定位根据 Web 页的其他元素来放置某个元素。

a．相对

b．绝对

c．内嵌

d．帧

16．____定位将元素放置在 Web 页内的指定位置上。

a．相对

b．绝对

c．内嵌

d．帧

17．与用于 CSS 定位的内嵌样式的 STYLE 属性一起使用的是哪种属性？

a．left 和 top

b．x 和 y

c．horizontal 和 vertical

d．x-axis 和 y-axis

18．Navigator 使用哪种专用的 HTML 标签对用于定位？

a．<SPAN>…</SPAN>

b．<DIV>…</DIV>

c．<LAYERS>…</LAYERS>

d．<POSITION>…</POSITION>

19．JavaScript 在 Navigator 下使用哪种方法来定位？

a．setAt()方法

b．top()和 bottom()方法

c．left()和 right()方法

d．moveTo()方法

7.2.9 练习

对下面的练习，从 Internet 或所用的图像程序中搜索所需的图像。创建每个程序以便它能够在常用的浏览器下运行：Navigator 或 Internet Explorer。

1．在 Student Disk 的 Tutorial.07 文件夹内包含了五幅篮球图像：从 basketball1.gif 到 basketball5.gif。将此图像制成动画让篮球在屏幕上上蹦下跳。在 Student Disk 的 Tutorial.07 文件夹内将文档存为 Dribble.html。

2．在 Student Disk 的 Tutorial.07 文件夹内包含了三幅雪花图像： snowflake1.gif、snowflake2.gif、snowflake3.gif。多次使用此图像来在 Web 页上创建暴风雪的效果。它从屏幕的顶端降落到底部。在 Student Disk 的 Tutorial.07 文件夹内将文档存为 Blizzard.html。

3．在 Student Disk 的 Tutorial.07 文件夹内包含了一幅袋鼠图像：kangaroo.gi。将它制成动画让袋鼠在屏幕上跳来跳去。在 Student Disk 的 Tutorial.07 文件夹内将文档保为 Kangaroo.html。

4．在 Student Disk 的 Tutorial.07 文件夹内包含了三幅鱼图像：fish1.gif、fish2.gif、fish3.gif。使用此图像在 Web 页内创建一个鱼缸。鱼在两个方向上游来游去。根据所需的图像数目使用此图像。在 Student Disk 的 Tutorial.07 文件夹内将文档存为 FishTank.html。

5．在 Student Disk 的 Tutorial.07 文件夹内包含了一幅赛车图像：racecar.gif。将此图像制成动画让跑车看起来像是在椭圆形的跑道上行驶。借用几个按钮让跑车以不同的速度行驶。在 Student Disk 的 Tutorial.07 文件夹内将文档存为 RaceTrack.html。

6．在 Student Disk 的 Tutorial.07 文件夹内包含了一幅飞机图像：airplane.gif。创建包含此飞机图像的 Web 页。创建几个控制飞机飞行的按钮。其中一个按钮用于起飞，一个用于着陆，另外一个用于循环飞行等。在 Student Disk 的 Tutorial.07 文件夹内将文档存为 Pilot.html。

第 8 章　Cookies 和安全

案例

随着创建的 JavaScript 程序越来越多，将发现 JavaScript 程序的缺限：不能保存访问网站者的信息。WebAdventure 公司的经理建议学习 Cookies 来保存访问者的信息。Cookies 是一小段信息，它能够被保存在用户的计算机内。然而，当了解了 Cookies 确实是保存在用户的计算机内时，又将关注安全问题，尤其是那些在新闻中得知的问题。因此，在 WebAdventure 的下一步任务是学习怎样创建 Cookies 和怎样实现 JavaScript 的安全特性。

浏览产品注册信息和主页程序

在本教程中，将着手两个独立的工程。第一个工程是修改在第 6 章创建的产品注册程序，将表单的内容保存在 Cookies 中，而不是保存在隐藏的表单域内。第二个工程是一个特定的 Navigator 程序，它改变用户的默认主页。不应该编写此类程序，因为它闯入了 JavaScript 程序的禁地。然而，创建此类程序是为了理解 JavaScript 安全是如何禁止篡改用户的系统程序的行为。将对创建的安全程序签署数字证书，它标明了 JavaScript 程序的编写者。不能够运行安全程序，因为必须首先创建测试证书。然而，可以浏览程序的代码。

浏览产品注册程序：

1. 在 Web 浏览器中打开 Data Disk 的 Tutorial.08 文件夹内的 Tutorial_8 ProductRegistration.html 文件。程序的功能与第 6 章中的相同，不同之处在于表单数据现在保存在本地计算机上的 Cookies 中，而不是保存在隐藏的表单域中。

2．关闭 Web 浏览器窗口。

3.在文本编辑器或 HTML 编辑器中打开 Tutorial_8CustomerInfo.html 文件。NextForm() 函数创建了保存在 savedData 变量中的字符串，它由一列相互连接的用于表单上的所有域的“名称=值”对组成。在创建了字符串之后，将它赋给 Document 对象的 cookie 属性。

4．关闭 Tutorial_8CustomerInfo.html。

浏览主页程序代码：

1． 在文本编辑器或 HTML 编辑器中打开 Tutorial8_HomePage.html 文件。注意在脚本节的 Archive 属性引用了 Tutorial8_HomePage.jar 文件。带有.jar 后缀的文件是一种特殊的

压缩文件，它包含了署名的脚本内容。还请注意<SCRIPT>标签和表单按钮包含了惟一的 ID 标签。Archive 和 ID 属性用于数字签署 JavaScript 程序。若运行程序并单击按钮，将会弹出对话框提示是同意还是拒绝读取个人信息。对话框提示来自 dongosselin 的 JavaScript 或 Java 小应用程序需要额外的特权。Tutorial8_HomePage.html 文件是一段签署过的具有名为 dongosselin 数字证书的脚本。要浏览证书，单击“ Certificate ”按钮。图 8-1 显示了 dongosselin 数字证书实例。

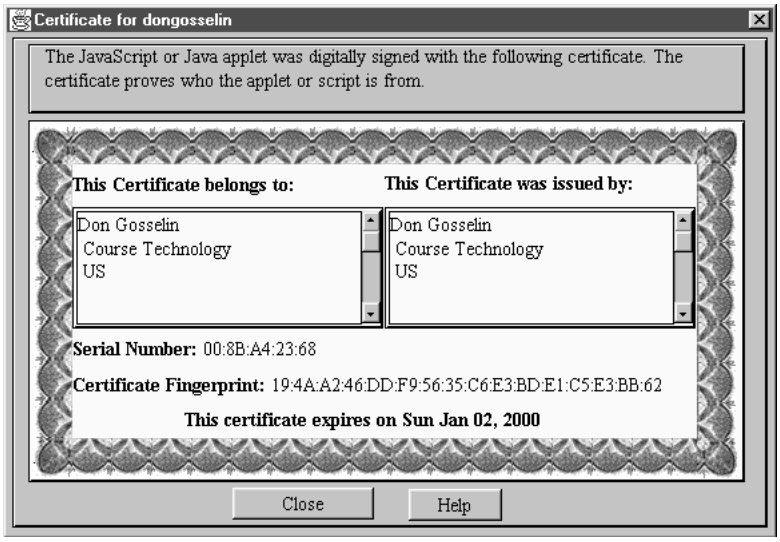


图 8-1 数字证书

2．关闭 Tutorial8_HomePage.html 文件。

8.1 状态信息和 Cookies

本节目标

在本节将学习：

- ✧ 状态信息
- ✧ String 对象
- ✧ 怎样保存带有查询字符串的状态信息
- ✧ 怎样创建和读取 Cookies

8.1.1 状态信息

超文本传输协议（HTTP）管理用于浏览 Web 的超文本链接和确保 Web 浏览器正确处

理和显示 Web 页内不同类型的内容。最初设计的 HTTP 是静态的，因为它不保存每次访问 Web 页的持续化数据。最初的 Web 的静态设计让早期的 Web 服务器能够快速地处理对 Web 页的请求，因为它们不需要记住不同用户的任何不同的需求。同样，浏览器也不需要任何特殊信息从服务器下载特殊 Web 页。尽管这种静态设计是有效的，但是它也有缺陷，因为 Web 服务器不能记住每个用户的信息。浏览器将每次 Web 页访问都作为全新的会话，而不管用户是否是在同一服务器上打开不同的 Web 页，是否切换到另一个不同的网址，还是完全关闭了他们的浏览器。这种设计妨碍了交互性并且限制了网址能够提供的个性化。现在，维护状态信息的理由包括：

- ✧ 根据用户的偏爱来定制每个 Web 页。
- ✧ 在浏览由多部分组成的表单时临时保存信息。
- ✧ 为返回到网址内的特殊位置做标记。
- ✧ 为商业网址保存订单信息购物车。
- ✧ 保存用户 ID 和密码。
- ✧ 跟踪用户访问网址次数的计数器。

用来维护状态信息的方法有数种。最常用的方法之一是隐藏表单域。另外两种维护状态信息的方法是查询字符串和 Cookies。为了了解查询字符串和 Cookies，将学习在第 6 章中创建的产品注册程序。产品注册程序由在 ProductRegistration.html 文件中创建的两个框架组成。底部框架用来在两个 Web 页（ProductInfo.html 和 CustomerInfo.html）之间切换。为了记录用户在 ProductInfo.html 和 CustomerInfo.html 之间浏览的状态信息，每个 Web 页上的表单的内容都被拷贝到 ProductRegistration.html 顶部框架的隐藏的表单域中。首先，修改 ProductRegistration 程序，使用查询字符串来维护状态信息。接着修改程序使用 Cookies 来维护状态信息。

8.1.2　String 对象

JavaScript 中的所有字符串的内容和字符串变量都能够用 String 对象来表示。String 对象包含了用来维护文本字符串的方法。表 8-1 列出了 String 对象常用的方法。

表 8-1　String 对象常用的方法

方　法	描　述
anchor（anchor 名称）	为文本字符串添加<ANCHOR>…</ANCHOR> 标签对
big()	为文本字符串添加<BIG>…</BIG> 标签对
blink()	为文本字符串添加<BLINK>…</BLINK> 标签对
bold()	为文本字符串添加<BOLD>…</BOLD> 标签对
charAt(index)	返回文本字符串内指定位置上的字符。若指定的位置大于字符串的长度，那么返回空
fixed()	为文本字符串添加<TT>…</TT> 标签对
fontcolor(color)	为文本字符串添加<FONT COLOR=*color*>…</FONT> 标签对

续表

方　法	描　述
fontsize(size)	为文本字符串添加<FONT SIZE=*size*>…</FONT> 标签对
indexOf(text, index)	返回 text 参数内的首字符在字符串中的位置编号
italics()	为文本字符串添加<I>…</I> 标签对
lastIndexOf(text, index)	返回 text 参数内中首字符的最后一个实例在字符串中的位置编号
link(href)	为文本字符串添加<A HREF=*URL*>…</A> 标签对
small()	为文本字符串添加<SMALL>…</SMALL> 标签对
split(separator)	根据指定的分隔符将文本字符串分成子串数组
strike()	为文本字符串添加<STRIKE>…</STRIKE> 标签对
sub()	为文本字符串添加_… 标签对
substring(starting index, ending index)	从 starting index 参数所指定的在字符串中位置编号开始，至 ending index 参数所指定在字符串中的位置编号结束，从字符串中提取文本
sup()	为文本字符串添加[…] 标签对
toLowerCase()	将指定的字符串转换为小写
toUpperCase()	将指定的字符串转换为大写

String 对象还包括了一个简单的属性——length 属性，它返回字符串中字符的个数。在字符串内容或 String 变量的后面添加句点和 String 对象的方法或 length 属性来维护字符串。注意可以使用下标 0 来引用在文本字符串内的首字符。例如，使用下标 0 来引用字符串“ JavaScript ”中的首字符 J，使用下标 1 来引用第二个字符 a，依此类推。为了使用 charAt() 方法返回字符串“ JavaScript ”内的首字符（J），使用语句"JavaScript".charAt(0);。

下面的代码使用了 String 对象的几个方法以及 length 属性。在每条语句后面的注释显示了每个方法或属性的结果。

```
var newString;
newString = "JavaScript".bold();
                                        // <B>JavaScript</B>
newString = "JavaScript".charAt(5);     // c
newString = "JavaScript".indexOf("S");  // 4
newString = "JavaScript".lastIndexOf("a");  // 3
newString = "JavaScript".substring(4,10);   // Script
newString = "JavaScript".toUpperCase();     // JAVASCRIPT
newString = newString.length;           // 10
```

回想一下可以使用简单的“ + ”和“ － ”操作符来维护字符串。例如，能够使用语句 myName = "Don" + "Gosselin"来连接文本字符串。此语句将字符串“ Don ”和“ Gosselin ”连接起来并将新字符串赋给变量 myName。能够使用“ + ”和“ += ”赋值运算符和 String 对象的方法来创建保存状态信息的文本。为了使用保存在长文本字符串内的状态信息，通

常必须解析长字符串。解析是从更大的字符串内提取出字符或子串。

图 8-2 显示使用 String 对象的数个方法、“ + ”赋值运算符以及 length 属性。图 8-3 显示了程序在浏览器中的输出结果。

```
<HTML>
<HEAD>
<TITLE>String Object Examples</TITLE>
</HEAD>
<BODY>
<PRE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
var myCity = "Boston";
document.writeln("Length: " + myCity.length);
// The following line converts the variable
// to <B>Boston</B>
document.writeln("Bolded variable: " + myCity.bold());
document.writeln("Character at 3: " + myCity.charAt(3));
document.writeln("Index of 's': " + myCity.indexOf('s'));
document.writeln("Substring 3, 6: " +
      myCity.substring(3,6));
document.writeln("San Francisco".toLowerCase());
document.writeln("San Francisco".toUpperCase());
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

图 8-2　String 对象实例

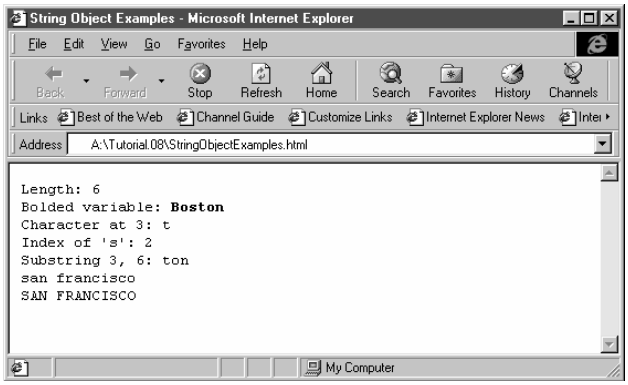


图 8-3　在浏览器中 String 对象实例

提示：使用 String 对象的方法和 length 属性能够在任何时候维护文本字符串。

8.1.3 使用查询字符串保存状态信息

一种用来保存用户访问 Web 页的信息的方法是在 URL 的后面添加查询字符串。查询字符串是一组添加在目标 URL 后的“名称=值”的集合，它由一个包含了一段或多段信息的文本字符串组成。可以使用查询字符串将信息从一个 Web 页传递给另一个 Web 页，如搜索规则。为了将信息从一个 Web 页传递给另一个 Web 页，请将查询字符串传递给 Web 页的 Location 对象的 search 属性。Location 对象的 search 属性包含了一个 URL 查询或搜索参数。为了使用在查询字符串内的数据，必须使用 String 对象的方法和 length 属性来维护字符串。

提示：若目标 Web 页是 CGI 脚本，那么查询字符串接着被赋给环境变量的 QUERY_STRING。

提示：Location 对象的 search 属性的名称来源于许多 Internet 搜索引擎使用查询字符串来保存搜索规则。

要创建查询字符串，请在 URL 后紧跟着添加问号（?），接着是所要保存的信息的“名称=值”。这种方法（将信息传递给另一个 Web 页）与将参数传递给函数或方法类似。每个“名称=值”通过查询串内的与（&）来分隔。下面的代码提供了一个<A>…</A>标签对的实例，其中包含了一个具有三对“名称=值”的查询字符串。

```
<A HREF="http://www.URL.com/TargetPage.html?firstName=Don
&lastName=Gosselin&occupation=writer">Link Text</A>
```

在 TargetPage.html 文件打开之后，在 Location 对象的 search 属性内能够访问查询字符串?fristName=Don&lastName=Gosselin&occupation=writer。

下一步，将修改在第 6 章创建的产品注册程序，将客户信息作为查询字符串传递而不是保存在隐藏的表单域内。

修改产品注册程序来将客户信息作为查询字符串传递而不是保存在隐藏的表单域内：

1．将文件 ProductRegistration.html、ProductInfo.html、CustomerInfo.html 和 TopFrame.html 从 Data Disk 的 Tutorial.06 文件夹拷贝到 Tutorial.08 文件夹内。

2．在文本编辑器或 HTML 编辑器内打开 TopFrame.html 文件。删除包含了隐藏的表单域的表单，接着保存和关闭文件。不再需要表单，因为数据将被保存在查询字符串内。

3．在文本编辑器或 HTML 编辑器内打开 CustomerInfo.html 文件。

4．使用下面的代码（它利用每个表单元素名和 savedData 变量的值来创建查询字符串）代替 nextForm()函数内的所有语句。每个表单元素名作为字面字符串输入，并且使用“+”和“+=”赋值运算符将它与每个元素的属性值串联起来。所有“名称=值”被串联为一个

字符串，并被赋给 savedData 变量。注意每对“名称=值”被“&”分隔开了。最后一条语句将问号和 savedData 查询字符串添加在 CustomerInfo.html URL 的后面。

```
var savedData;
savedData = "name=" +
        document.customerInfo.name.value;
savedData += "&address=" +
        document.customerInfo.address.value;
savedData += "&city=" +
        document.customerInfo.city.value;
savedData += "&state=" +
        document.customerInfo.state.value;
savedData += "&zip=" +
        document.customerInfo.zip.value;
savedData += "&email=" +
        document.customerInfo.email.value;
savedData += "&password=" +
        document.customerInfo.password.value;
if (document.customerInfo.elements [7 ].checked == true)
        savedData += "&platform=win95-98";
else if (
        document.customerInfo.elements [8 ].checked == true)
        savedData += "&platform=winnt";
else if (
        document.customerInfo.elements [9 ].checked == true)
        savedData += "&platform=UNIX";
else if (
        document.customerInfo.elements [10 ].checked == true)
        savedData += "&platform=mac";
if (document.customerInfo.wp.checked == true)
        savedData += "&wp=true";
if (document.customerInfo.ss.checked == true)
        savedData += "&ss=true";
if (document.customerInf o.db.checked == true)
        savedData += "&db=true";
if (document.customerInfo.gr.checked == true)
        savedData += "&gr=true";
if (document.customerInfo.pr.checked == true)
        savedData += "&pr=true";
if (document.customerInfo.location.options [0 ].selected
        == true)
        savedData += "&location=work";
```

```
else if (
        document.customerInfo.location.options [1 ].selected
        == true)
        savedData += "&location=school";
else if (
        document.customerInfo.location.options [2 ].selected
        == true)
        savedData += "&location=home";
else if (
        document.customerInfo.location.options [3 ].selected
        == true)
        savedData += "&location=home_office";
savedData += "&comments=" +
document.customerInfo.comments.value;
location.href="ProductInfo.html" + "?" + savedData;
```

5．保存并关闭 CustomerInfo.html。

下一步将修改 ProductInfo.html 文件，其中包含了用来将表单数据传递给镜像服务器的 Submit 按钮。在此练习中，将查询字符串添加在 URL 的后面而不将数据提交给服务器。因此，需要修改 submitForm()函数将查询字符串添加在镜像 CGI 脚本 URL 的后面。在开发环境下，需要编写 CGI 脚本来进一步处理在查询字符串内的“名称=值”对。

修改 ProductInfo.html 文件：

1．在文本编辑器或 HTML 编辑器内打开 ProductInfo.html 文件。

2．删除在 subitForm()函数内的所有语句。接着，添加下面的代码，它创建了新的 savedData 变量并将 ProductInfo.html 表单域添加在 Location 对象的 search 属性内的查询字符串的后面。

```
var savedData = location.search;
savedData += "&serial=" +
        document.productInfo.serial.value;
savedData += "&date=" +
        document.productInfo.date.value;
if (document.productInfo.elements [2 ].checked == true)
        savedData += "&where=retail";
else if (
        document.productInfo.elements [3 ].checked == true)
        savedData += "&where=catalog_mail";
else if (
        document.productInfo.elements [4 ].checked == true)
        savedData += "&where=internet";
else if (
```

```
document.productInfo.elements [5 ].checked == true)
savedData += "&where=other";
```

3．按下回车键并输入//location.href="ProcessOrder.cgi" + "? " + savedData;。确保在改变 Location 对象的 href 属性的语句的前面包含两个斜杠。这些斜杠将语句变为不执行的注释。此练习演示了将查询字符串添加在任何类型的 URL 的后面，但是 ProcessOrder.cgi 程序并不存在。因为不存在程序 ProcessOrder.cgi，所以若在不将改变 Location 对象的 href 属性的语句注释掉的情况下提交表单，将会接收到错误消息。

4．再次按下回车键，并添加语句 return false;来禁止提交表单。

5．保存 ProductInfo.html。在打开文件和运行程序之前，需要添加解析查询串的代码，接下来将学习它。

解析字符串

为了让 Web 页使用查询字符串内的信息，必须首先结合使用 String 对象的方法和 String 对象的 length 属性来解析字符串。第一项任务是使用 substring()方法和 length 属性来删除在查询字符串起始处的问号。substring()方法接收两个参数：起始下标和终止下标。字符串中的首字符下标为 0，与数组中的元素类似。因为想删除字符串的首字符（问号），它的下标为 0，所以将 1 作为起始下标。至于结束下标，将使用 length 属性，它通知 substring()方法包括字符串的剩余部分。下面的代码将 Location 对象的 search 属性赋给 queryData 变量并使用 substring()方法和 length 属性来删除起始问号。

```
// Assigns the query string to the queryData variable
var queryData = location.search;
// Removes the opening question mark from the string
queryData = queryData.substring(1, queryData.length);
```

下一步是使用 split()方法将 queryData 变量内的每对信息片断转换成数组元素。split()方法只接收一个参数，它表示用来分隔字符串内的每对信息片断的字符。使用“ & ”，因为正是它分隔查询字符串内的“ 名称=值 ”对。然而，记住能够在任何的字符处分隔字符串。将 queryData 变量内的信息转换为 queryArray 数组的代码如下：

```
// splits queryData into an array
var queryArray = queryData.split("&");
```

图 8-4 显示了解析程序的全部代码。

下一步，将解析 ProductInfo.html 查询字符串中的“ 名称=值 ”对并在警告对话框中显示它们：

1．返回到文本编辑器或 HTML 编辑器中的 ProductInfo.html 文件。

2. 在 submitForm() 函数内的 return false; 语句前插入语句 savedData= savedData.substring(1, savedData.length);。此语句使用 substring()方法来删除查询字符串内的

问号。

```
<HTML>
<HEAD>
<TITLE>Parsing Query Strings</TITLE>
</HEAD>
<BODY>
<PRE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
// Assigns the query string to the queryData variable
var queryData = location.source;
// Removes the opening question mark from the string
queryData = queryData.substring(1, queryData.length);
// splits queryData into an array
var queryArray = queryData.split("&");
document.writeln(queryArray [0]);
document.writeln(queryArray [1]);
document.writeln(queryArray [2]);
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

图 8-4 解析程序

3．按下回车键并输入 var dataArray = savedData.split("&");将查询字符串分隔为 dataArray[]数组。

4．添加下面的语句，它使用 for 循环在警告对话框内显示 dataArray[]的内容。警告对话框中的每对“名称=值”对由换行符（\n）分隔。

```
var messageString = "";
for (var i = 0; i < dataArray.length; ++i) {
        messageString += dataArray [i ] + "\n";
}
alert(messageString);
```

5．保存并关闭 ProductInfo.html 文件，接着在浏览器中打开 ProductRegistration.html 文件。填充客户信息表单中的域并单击“Submit Query”按钮。两个表单的“名称=值”对将被显示在警告对话框中。

6．关闭 Web 浏览器窗口。

8.1.4　使用 Cookies 保存状态信息

查询字符串不能永久地维护状态信息——包含在字符串中的信息仅能够在 Web 页的当前会话中访问。在 Web 所读取查询字符串的 Web 页关闭之后，查询字符串将被丢失。隐藏表单域也能在 Web 页之间维护信息，但是在读取隐藏表单域的 Web 页关闭之后它们所包含的数据也将被丢失。能够使用 CGI 脚本来保存查询字符串或隐藏的表单域的内容，但是此方法需要单独地基于服务器的程序。为了能够在当前的 Web 页会话之外保存状态信息，Netscape 创建了 Cookies。魔术般的 Cookies 是众多的小段信息，它们被 Web 服务器以文本文件的形式保存在用户的计算机上。每次 Web 客户访问 Web 服务器，所请求页面保存的 Cookies 被客户传递给服务器。接着服务器使用 Cookies 为客户提供个性化 Web 页。最初创建 Cookies 是为了与 CGI 一起使用，现在普遍地被 JavaScript 使用。考虑下面的情形：访问一个网址并被要求在提示对话框中或文本域中输入用户名。以后每次访问相同的网址，不管是在相同的浏览会话中还是在数天或数周之后的不同的浏览会话中，都将使用已输入的用户名。Web 页通过将它保存在计算机上的 Cookies 中来记住此信息。另一个已经看到过的典型的 Cookies 实例是计算每个用户已经访问某个网址的次数的计数器。

提示：此教程仅讨论与 JavaScript 相关的 Cookies。Cookies 的全部技术规范能够在线访问 http://www.netscape.com/newsref/std/cookie_spec.html。

Cookies 可以是临时的也可以是永久性的。临时 Cookies 仅能在当前的浏览会话中访问。永久性的 Cookies 在当前浏览会话之后仍能够访问，它们以文本文件的形式保存在客户计算机上。在本节中，将创建永久的和临时的 Cookies。

在 Windows 系统下的 Navigator 中，Cookies 被保存在 Navigator 目录下的 cookies.txt 文件中。Macintosh 系统下的 Navigator 将 Cookies 保存在 Netscape 文件夹内的 MagicCookie 文件中。Internet Explorer cookies 被包含在 Windows 系统目录内的 cookie 目录下的每个文件中。

提示：请不要打开和修改 cookie 文件，因为可能在不经意之间删除或改变了 Web 服务器所保存的重要信息。

使用 Cookies 具有一系列的限制。在用户的计算机上每个服务器或域仅能够保存最多 20 个 Cookies。另外，每个浏览器的总 Cookies 数不能够超过 300 个，并且 Cookie 的最大尺寸是 4k。若超过了这些限制，那么 Web 浏览器可能删除旧的 Cookies。

创建 Cookies

使用 Document 对象的 cookie 属性以“名称=值”对的方式来创建 Cookies，它与查询字符串所使用的“名称=值”对的方式相同。Document 对象的 cookie 属性格式是：document.cookie=name+value;。创建 cookie 属性必须包括一个必须的 name 属性和四个可选属性：expires 属性、path 属性、domain 属性和 secure 属性。

name 属性　cookie 的“名称=值”对参数是 cookie 属性惟一必须的参数，仅利用“名称=值”对参数创建的 Cookies 是短暂的，或临时的，因为它们仅能在当前的浏览会话中访问。下面的代码使用 firstName=Don“名称=值”对创建了一个 cookie：

```
document.cookie = "firstName=" + "Don";
```

Document 对象的 cookie 属性可能令人感到困惑。对其他的 JavaScript 属性，将新值赋给属性将代替旧值。相比之下，将新值赋给 cookie 属性将创建一系列的 cookie，而不仅仅是代替上一次的值。下面的实例创建了一系列的 cookie：

```
document.cookie = "firstName=" + "Don";
document.cookie = "lastName=" + "Gosselin";
document.cookie = "occupation=" + "writer";
```

还能够使用一条 document.cookie 语句来创建一系列的 cookie，它通过分号来分隔“名称=值”对。若使用分号创建一系列的 cookie，请确保将分隔“名称=值”对的分号放置在文本字符串内，否则它们将被解释为 JavaScript 语句的结束。

Cookies 本身不能包括分号或其他特殊的字符，如逗号或空格。Cookies 不能包括特殊的字符是因为在 Web 浏览器和 Web 服务器之间使用 HTTP 来传递它们。HTTP 不允许某个非字母和数字的字符以它们自身的格式被传递。因此，在将它们赋给 cookie 属性之前，对文本进行编码是一个不错的习惯。编码涉及到将文本字符串中的特殊字符转换成相应的十六进制的 ASCII 值，前面跟着百分号。例如，20 是空格字符的十六进制 ASCII 值，21 是惊叹号（！）的十六进制 ASCII 值。在 URL 编码格式中，每个退格字符用%20 代替，每个惊叹号用%21 代替。在编码之后，foods=My favorite food is pizza!字符串的内容将转换为 foods=My%20favorite%20food%20is%20pizza%21。

在 JavaScript 中使用 escape()方法来编码文本字符串。escape()方法的语法是 escape(text);。当读取 cookie 或其他以 escape()方法编码的文本字符串时，必须首先用 unescape()方法进行解码。unescape()方法的语法是 unescape(text);。下面的代码在 cookieList 变量中创建了一列 cookie。接着变量用 escape()方法进行编码并将它赋给 Document 对象的 cookie 属性。

```
var cookieList = "firstName=" + "Don"
      + ";lastName=" + "Gosselin"
      + ";occupation=" + "writer";
document.cookie = escape(cookieList);
```

下一步，将修改 ProductRegistration.html 文件以便将客户信息保存在临时 Cookies 中。

修改 ProductRegistration.html 文件以便将客户信息保存在临时 Cookies 中：

1．在文本编辑器或 HTML 编辑器中打开 CustomerInfo.html 文件。

2．在创建 savedData 变量的每一行上，用“ ; ”替换“ & ”。总共需要替换 20 个符号。

3．在 location.href 语句的前面添加语句 document.cookie = escape(savedData);，它将 savedData 字符串赋给当前文档的 cookie。在将它赋给 cookie 属性之前使用 escape()方法对 savedData 变量进行编码。

4．删除 location.href 语句中添加查询字符串的那部分，之后它为 location.href = "ProductInfo.html";。

5．保存并关闭 CustomerInfo.html 文件。

下一步，修改 ProductInfo.html 文件将其内的域添加给 cookie。

修改 ProductInfo.html 文件：

1．在文本编辑器或 HTML 编辑器中打开 ProductInfo.html 文件。

2．使用下面的语句来替换 submitForm()函数内的前 5 条语句，它将 ProductInfo.html 表单域添加给 cookie。

```
document.cookie = "serial=" +
        document.productInfo.serial.value;
document.cookie = "date=" +
        document.productInfo.date.value;
document.cookie = "where=" +
        document.productInfo.where.value;
```

3．保存并关闭 ProductInfo.html 文件。尽管能够打开表单和输入数据，但是在学会怎样读取 cookie 之前，将不能够使用在表单之间传递的数据。在学习怎样读取 cookie 之前，将先学习 cookie 的其他选项。

expires 属性　直到现在仅讨论了在当前浏览会话结束之后就终止的临时的 Cookies。为了在当前浏览会话之外持续化 cookie，必须使用 expires 属性。cookie 属性中的 expires 属性决定了在 cookie 被删除之前它保存在客户系统内的时间。不具有 expires 属性的 cookie 仅能在当前浏览会话内访问。将 expires 属性赋给 cookie 属性的语法是 expires=date，以及相关的“ 名称=值 ”对。“ 名称=值 ”对和 expires=date 对使用“ ; ”分隔，与分隔多个“ 名称=值 ”对一样。expires 属性的 date 部分必须是如下格林尼治标准时间格式的文本字符串：

```
Weekday Mon DD HH:MM:SS Time Zone YYYY
```

下面是一个格林尼治标准时间的实例：

```
Mon Dec 27 14:15:18 PST 1999
```

能够手工地输入格林尼治标准时间格式的字符串，或使用 Date 对象来创建字符串。Date 对象用来处理日期和时间。创建新 Date 对象将把本地计算机的日期和时间拷贝到指定的对象名中。接着能够使用 Date 对象的方法来维护变量中的日期和时间。注意变量中的日期和

时间并不像时钟那样随着时间的流逝而不断地更新。相反，Date 对象包含的日期和时间是静态的，它是 JavaScript 代码执行时的时间。使用语法 var variable = new Date();来创建 Date 对象的新的实例。表 8-2 列出了 Date 对象常用的方法。

表 8-2　常用的 Date 对象方法

方　法	描　述
getDate()	返回 Date 对象的日期
getDay()	返回 Date 对象的日
getFullYear()	以 4 个数字的格式返回 Date 对象的年份
getHours()	返回 Date 对象的小时
getMinutes()	返回 Date 对象的分钟数
getMonth()	返回 Date 对象的月份
getSeconds()	返回 Date 对象的秒数
getTime()	返回 Date 对象的时间
setDate()	设置 Date 对象的日期
setFullYear()	用 4 位数字设置 Date 对象的年份
setHours()	设置 Date 对象的小时数
setMinutes()	设置 Date 对象的分钟数
setMonth()	设置 Date 对象的月份
setSeconds()	设置 Date 对象的秒数
setTime()	设置 Date 对象的时间
toGMTString()	将 Date 对象转换为 GMT 时区的字符串
ToLocalString()	将 Date 对象转换为本地时区的字符串

Date 对象的每部分，如日、月份、年份等，都能够使用 Date 对象的方法来检索和修改。例如，若创建新的 Date 对象，使用语句 var myDate = new Date();，那么能够使用语句 myDate.getDate();，仅检索 myDate 对象中的日期部分。

为了将 Date 对象与 expires 属性一起使用，使用 Date 对象的 set()和 get()方法来设置 cookie 有效的时间。下面的实例使用 setDate()和 getDate()方法修改 myDate 对象的日期部分。注意能够在其他 Date 对象方法内内嵌 Date 对象的方法。在实例中，setDate()方法设置了 myDate 的日期部分，它使用 getDate()检索到日期，并对它加 7 为日期增加一周。

```
myDate.setDate(myDate.getDate() + 7);
```

下面的代码创建新的 cookie 并将从现在起一年后过期的日期赋给它。当 expires 属性被赋给 cookie 的属性时，Date 对象使用 toGMTString()方法来保证日期是 GMT 格式。

```
var expiresDate = new Date();
document.cookie = escape("firstName=Don" + "; expires="
```

```
    + expiresDate.setFullYear(expiresDate.getFullYear()
      + 1));
```

为了删除 cookie，重新为它命名，并为 cookie 属性随便赋给一个值，将它的过期日期设置为过去的某个时间。下面的代码将 firstName cookie 的 expires 属性设置为一年前来删除它。

```
document.cookie = escape("firstName=” + "delete" +
"; expires=" + expiresDate.setFullYear(
expiresDate.getFullYear() -1));
```

下一步，将创建一个在持续化 cookie 中保存用户名和喜欢的背景颜色的文档。在此练习中，将首先创建引用文档体内表单的函数。在创建函数之后，接着将创建文档的体和表单部分。

创建在持续化 cookie 中保存用户名和喜欢的背景颜色的文件：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．输入文档的<HTML>、<HEAD>和<SCRIPT>段的起始部分。

```
<HTML>
<HEAD>
<TITLE>Personal Preferences</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

3．添加下面的 setCookie()函数，它创建了一个包含了用户名和偏爱颜色（它们的值来自 user_name 和 user_color 表单域）的 cookie。并设置 cookie 一年后过期。

```
function setCookie() {
        var expiresDate = new Date();
        var userCookies = "user_name=" +
                document.user_prefs.user_name.value
                + ";user_color=" +
                        document.user_prefs.user_color.value
                + ";expires=" +
                expiresDate.setFullYear(
                        expiresDate.getFullYear() + 1);
        document.cookie = escape(userCookies);
        alert(
        "Your name and favorite color have been saved in a
cookie.");
}
```

4．输入<SCRIPT>和<HEAD>标签的结束部分以及<BODY>标签的起始部分。

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
<BODY>
```

5．添加下面的表单，用户将使用它在 cookie 中保存他们的姓名和偏爱的颜色。按钮使用 onClick 事件处理器来调用头段内的函数。

```
<FORM NAME="user_prefs">
Name  <INPUT TYPE="text" NAME="user_name"><BR>
Color  <INPUT TYPE="text" NAME="user_color"><P>
<INPUT TYPE="button" VALUE=" Set Cookie "
        onClick="setCookie();"><P>
</FORM>
```

6．添加下面的代码结束<BODY>和<HTML>标签。

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

7．在 Data Disk 的 Tutorial.08 文件夹内将文件存为 PersonalPrefs.html，接着在浏览器中打开此文件。输入姓名和偏爱的颜色，单击“ Set Cookie ”按钮。尽管文件保存了 cookie，但是在能够编写返回到此 Web 页时就显示用户的首选选项的代码之前，需要学习怎样使用 JavaScript 来读取 cookie。在此节的后面部分将学习怎样读取 Cookies。

path 属性　该属性确定了 cookie 能够被服务器上其他 Web 页的访问性。接着 path 属性被赋给 cookie 属性。默认时，与 cookie 在相同目录下的所有 Web 页都能够访问。然而若指定了路径，那么在指定路径下与指定路径下的所有子目录下的所有 Web 页都能够访问。例如，下面的语句让 firstName cookie 能够被 MyFiles 目录下或它的任何子目录下的所有 Web 页访问：

```
document.cookie = "firstName=Don" + "; path=/MyFiles");
```

为了让 cookie 能够被服务器上的所有目录访问，请使用斜杠来表示根目录，如下面的实例中：

```
document.cookie = "firstName=Don" + "; path=/");
```

domain 属性　使用 path 属性允许 Cookies 能够跨服务器共享。然而，某些网址非常大并且使用了一系列的服务器。domain 属性能够被用于相同域内的跨多服务器共享 Cookies。

注意不能在域外共享 Cookies。使用语法 domain=域名和相关的“名称=值”对将 domain 属性赋给 cookie 属性。例如，若 Web 服务器 programming.gosselin.com 需要与 Web 服务器 writing.gosselin.com 共享 Cookies，那么 programming.gosselin.com 设置的 Cookies 的 domain 属性应该被设置为 gosselin.com。通过此方法，programming.gosselin.com 创建的 Cookies 能够被 writing.gosselin.com 和 gosselin.com 域内的所有其他服务器访问。

若域以 7 个顶级域标识符（.com、.edu、.net、.org、.gov、.mil 或.int）之一结束，那么需要在 gosselin.com 域内使用两个句点。例如，为了使用 gosselin.com 的 domain 属性，那么请使用语句 domain=.gosselin.com 因为.com 是 7 个顶级标识符之一。若域不是以 7 个顶级标识符之一结束，那么必须使用三个句点并包括一个子域。例如，若域以.ca（Canada）结束，那么必须使用如下的语句 domain=.子域.域.ca。

下面的代码演示了怎样使 programming.gosselin.com 上的 cookie 能够被 gosselin.com 域内的所有服务器访问：

```
document.cookie = "firstName=Don" + ";
        domain=.gosselin.com";
```

secure 属性　用标准 Internet 连接来传输敏感信息并不总是安全的。不道德的人们在线窃取私有信息（如信用卡号、口令、社会保险号和其他类型的私有信息）是可能的。为了保护在 Internet 上传输的私有数据，Netscape 开发了安全套接字层（SSL）来对数据加密并在安全的连接上传输它。支持 SSL 的网址通常以 HTTPS 而不是 HTTP 开始。secure 属性指示 Cookies 只能在安全的 Internet 连接（使用 HTTPS 或其他安全协议）上传输。通常在使用客户端 JavaScript 时，将忽略 secure 属性。然而，若想使用此属性，将值为 true 或 false 布尔值的此属性赋给 cookie 属性，同时赋给相关的“名称=值”对，语法为 secure=Boolean value。例如，为了 cookie 激活 secure 属性，使用类似于下面的语句：

```
document.cookie = "firstName=Don" + "; secure=true");
```

读取 Cookies

在 Document 对象的 cookie 属性内能够使用特殊 Web 页内的 Cookies。到目前，已经保存了临时的和持续化的 Cookies。下一步将学习怎样检索保存的 cookie 值。Cookies 由一个连续的字符串组成，在能够使用它所包含的数据之前，必须解析它。这意味着必须使用 String 对象的方法来解析每个“名称=值”对。另外，若 Cookies 用 escape()函数编码过，那么在解析它们之前，必须使用 unescape()函数将它们解码。为了理解从 Cookies 中解析数据涉及到哪些方面，下面的代码创建了三个编过码的 Cookies，接着从 cookie 属性中读取它们并将它们解码。接着使用 split()方法将每对“名称=值”对拷贝到 cookieArray 数组的元素中。

```
var secondCookie = "city=Boston";
```

```
var firstCookie = "team=Red Sox";
var thirdCookie = "sport=baseball ”;
document.cookie = escape(firstCookie + ";"
        + secondCookie + ";" + thirdCookie);
var cookieString = unescape(document.cookie);
var cookieArray = cookieString.split(";");
```

在将 Cookies 分割为每个数组元素之后，还需要确定哪个 cookie 保存了所需要的值。下面的 for 循环遍历数组中的每个元素并使用 if 语句和数个字符串方法来检测每对“ 名称=值 ”对的名称部分是否等于 team。if 语句中的条件表达式使用 substring()方法来返回 yourTeam 变量中“ 名称=值 ”对内的名称部分。substring()方法的第一个参数指定了子串的起始点（首字符 0）。substring()方法的第二个参数是添加在 yourTeam 变量后的 indexOf()方法，它返回等号的下标。若子串等于 team，那么使用 break 语句结束 for 循环，并将 Your team is the 文本串和“ 名称=值 ”对的值部分一起写向浏览器。返回“ 名称=值 ”对内值部分的语句也使用了 substring()方法和 indexOf()方法。然而，这次第一个参数指定子串从等号的下标加 1 开始，它是等号后的第一个字符。substring()方法的第二个参数指定了子串的终止点，它是数据变量的长度。

```
var yourTeam;
for (var count = 0; count < 4; ++count) {
        yourTeam = cookieArray [count ];
        if (yourTeam.substring(0,yourTeam.indexOf("="))
                == "team") {
                document.writeln("Your team is the "
                + yourTeam.substring(yourTeam.indexOf("=") + 1,
                        yourTeam.length));
                break;
        }
}
```

前面的代码难于理解。若对理解怎样维护字符串存在困难，请尝试使用不同的 String 对象方法，并观察它们的结果。使用字符串方法解析 cookie 是从 cookie 的长字符串中提取每个信息片断的惟一方法。

下一步，将创建一个读取和显示 ProductRegistration.html 文件所创建的 Cookies 的内容的函数：

1．在文本编辑器或 HTML 编辑器中打开 ProductInfo.html。

2．在 savedData=savedData.substring(1,savedData.length); 语句前为 submitForm()函数添加下面的语句，它将未解码的 cookie 值赋给 savedData 变量。

```
var savedData = unescape(document.cookie);
```

3．在 submitForm()函数内删除 savedData = savedData.substring(1, savedData.length);语句，它从查询字符串的起始处删除问号。因为 Cookies 中未包括起始的问号，所以不再需要此语句。

4．在将 savedData 变量分割为 dataArray 数组的语句内，将 split()函数方法参数内的“ & ”改为分号，因此语句应为 var dataArray = savedData.split(";");。

5. 保存并关闭 ProductInfo.html 文件，接着在 Web 浏览器内打开 Product Registration.html 文件。填充 Customer Information 表单内的域并单击“ Next ”按钮，接着填充 Product Information 表单内的域并单击“ Submit Query ”按钮。两个表单内的“ 名称=值 ”对应该显示在警告对话框内，与查询字符串所显示的一样。

6．关闭 Web 浏览器窗口。

下一步，将修改 PersonalPrefs.html 文件以便读取存储在 Cookies 内的用户的个人选择：

1．在文本编辑器或 HTML 编辑器内打开 PersonalPrefs.html。

2．为脚本段添加下面的函数，在<BODY>标签内的文档的 onLoad 事件处理器将调用它。if 语句判断在文档的 cookie 内是否保存了任何值。若未保存任何值，那么将跳过读取 cookie 的语句。若 cookie 包含了值，那么警告对话框将向用户显示名称，并将文档的背景颜色设置为用户偏爱的颜色。两条消息片断都保存在使用表单元素所创建的持续化 cookie 中。首先，使用 unescape()方法将 cookie 解码并将它赋给 curCookie 变量。接着，函数使用 indexOf()和 substring()方法以及 String 对象的 length 属性来确定每对“ 名称=值 ”对中的值部分。

```
function getUserPrefs() {
        var curCookie = unescape(document.cookie);
        if (curCookie != "") {
        var start = curCookie.indexOf("user_name")
                + "user_name".length + 1;
        var end = curCookie.indexOf(";", start);
        var user_name = curCookie.substring(start, end);
        alert("Welcome " + user_name
                + ". Click OK and I will set your favorite color.");
        start = curCookie.indexOf("user_color ")
                + "user_color".length + 1;
        end = curCookie.indexOf(";", start);
        var user_color = curCookie.substring(start, end);
        document.bgColor = user_color;
        }
}
```

3．为<BODY>标签添加 onLoad 事件调用 getUserPrefs()函数，如下：<BODY onLoad="getUserPrefs();">。

4．保存并关闭 PersonalPrefs.html，接着在 Web 浏览器中打开此文件。在文本框中输入姓名和偏爱的颜色，接着单击“ Set Cookie ”按钮。将弹出一个警告对话框告诉已保存在 cookie 中的姓名和颜色。单击“ 确定 ”关闭警告对话框，接着重新载入或刷新此页面。先前输入的姓名将出现在警告对话框中。在关闭警告对话框之后，文档的背景色应变为所选的颜色。

5．关闭 Web 浏览器窗口。

8.1.5 总结

- ✧ 状态信息指任何保存的与前面所访问网址相关的信息。
- ✧ String 对象包含了维护文本字符串的方法和属性。
- ✧ 解析是指从大的字符串中提取字符或子串的行为。
- ✧ 查询字符串是一组添加在目标 URL 后的“ 名称=值 ”对，它由一条包含了一条或多条信息片断的文本字符串组成。
- ✧ Location 对象的 search 属性包含了 URL 查询或搜索参数。
- ✧ 为了让 Web 页使用查询字符串内的信息，必须首先结合使用 String 对象的多个方法和 length 属性来解析字符串。
- ✧ Cookies 或神奇的 Cookies 是以文本形式由服务器保存在用户计算机上的与用户相关的小信息片断。
- ✧ Cookies 可以是临时的或持续化的。仅能够在当前浏览器会话中使用临时 Cookies。持续化 Cookies 在当前浏览器会话之后也能使用，它们以文本文件的形式保存在客户计算机上。
- ✧ 以“ 名称=值 ”对使用 Document 对象的 cookie 属性来创建 Cookies 与在查询字符串中使用“ 名称=值 ”对相同。
- ✧ cookie 属性仅需的属性是 name 属性，它指定了 cookie 的“ 名称=值 ”对。
- ✧ JavaScript 内使用 escape()方法来对文本字符串编码。
- ✧ 在读取利用 escape()方法编过码的 cookie 或其他文本字符串时，必须首先使用 unescape()方法对它解码。
- ✧ cookie 的 expires 属性决定了 cookie 在被删除之前，它在客户的系统上保留的时间。
- ✧ Date 对象被用来维护日期和时间。
- ✧ path 属性决定了 cookie 被服务器上其他 Web 页的可访问性。
- ✧ domain 属性用于在同一域内跨多服务器共享 Cookies。
- ✧ secure 属性指明了 cookie 仅能在使用 HTTP 或其他安全协议的安全的 Internet 连接上传输。
- ✧ 可以用 Document 对象的 cookie 属性来访问特殊 Web 页的 Cookies，它们由一个连续的字符串组成。在使用它们所包含的数据之前，必须解析它们。

8.1.6　问题

1．用来保存前面已访问过的网址的信息被称为____信息。
a．HTTP
b．客户端
c．状态
d．预先
2．用来维护文本字符串的方法和属性被存储在____对象中。
a．String
b．Text
c．TextString
d．StringText
3．____表示从大的字符串中提取字符或子串的行为。
a．Extrapolating
b．解析
c．摘录
d．连接
4．下面返回字符串中的字符数的语法哪个是正确的？
a．text_string.length;
b．text_string.characters();
c．text_string.numChars();
d．text_string.latIndex;
5．下面将字符串转换为小写的语法哪个是正确的？
a．text_string.down();
b．text_string.lower();
c．text_string.LowerCase();
d．text_string.toLowerCase ();
6．查询字符串中的每一项以哪种格式添加在目标 URL 的后面？
a．以逗号分割的格式
b．以&valuename 对
c．以“名称=值”对
d．以名称、值、长度格式
7．哪个字符被用来将查询字符串添加在 URL 之后？
a．?
b．&
c．$

d．%

8．查询字符串被保存在目标 URL 中的何处？

a．Document 对象的 queryString 属性

b．Window 对象的 query 属性

c．Location 对象的 search 属性

d．查询字符串不保存在目标 URL 中

9．哪个字符被用来分隔查询字符串中的项？

a．?

b．&

c．$

d．%

10．____方法被用来将查询字符串中的每个信息分隔为数组元素。

a．divide()

b．toArray()

c．queryToArray()

d．split()

11．Cookies 被保存在何处？

a．Web 客户上

b．Web 服务器上

c．HTTP 连接内

d．从不保存 Cookies

12．下面创建 cookie 的语法哪条是正确的？

a．cookie = name + value ;

b．document.cookie = name + value ;

c．window.cookie = name + value ;

d．this.cookie = name + value ;

13．下面哪个是 cookie 属性中惟一必需的属性？

a．name

b．domain

c．expires

d．path

14．在 JavaScript 中使用____方法来编码文本字符串。

a．protect()

b．encode()

c．encrypt()

d．escape()

15．不具有 expires 属性的 Cookies 被称为____。

a．短暂的
b．临时的
c．永久的
d．持续化的

16．expires 属性的日期部分必须是哪种格式的文本字符串？
a．DD-Mon-YY
b．HH:MM:SS
c．格林尼治时间（GMT）
d．它可以是任何时间格式

17．新 Date 对象中的日期和时间何时被更新？
a．每毫秒
b．每秒
c．根据它的时间间隔参数
d．从不

18．让服务器上的其他 Web 页能够访问 cookie 取决于____属性。
a．path
b．directory
c．system
d．server

19．哪种属性被用来在域外共享 Cookies?
a．domain
b．share
c．secure
d．在域外不能共享 Cookies

20．____属性指示仅在使用 HTTPS 或其他安全协议的安全的 Internet 连接上传输 cookie。
a．domain
b．share
c．secure
d．expires

8.1.7　练习

1．创建一个能够保存和读取包含了用户名和访问网址次数的 cookie 的文档。在任何时候用户访问网址时，在警告对话框中显示 Cookies，并将计数器 cookie 递增 1，接着重新将计数器的过期日期设置为一年后（从现在开始）。在 Data Disk 的 Tutorial.08 文件夹内将文档存为 Counter.html。

2．创建一个具有“ nag ”计数器（提醒用户注册）的文档。将计数器保存在 cookie 中，每当用户访问网址五次时就显示一条消息提醒用户注册。在文档体内创建一个表单，它包含了用于用户名和 E-mail 地址的文本框以及 Registration 按钮。在用户填写文本框和单击“ Registration ”按钮之后，删除 nag 计数器 cookie 并将它用包含了用户名和 E-mail 地址的 cookie 代替。在注册之后，任何时候用户访问网址，在警告对话框中显示姓名和 E-mail 地址 cookie。在 Data Disk 的 Tutorial.08 文件夹内将文档存为 NagCounter.html。

3．创建一个提示用户输入用户名和密码的文档，接着将信息保存在 Cookies 中。在用户再次访问网址时，提示用户输入保存的用户名和密码。若用户未在三次之内输入正确的信息，让用户输入新的用户名和密码。在 Data Disk 的 Tutorial.08 文件夹内将文档存为 Password.html。

4．创建一个包含了表单的文档，表单内具有代表不同颜色（如红、蓝、黄等）的按钮。在用户单击表单上的颜色按钮时，将文档的颜色变为相应的颜色，并将它保存在 cookie 中。在任何时候用户重新访问此 Web 页时，将背景重设置为 cookie 中所保存的颜色。在 Data Disk 的 Tutorial.08 文件夹内将文档存为 FavoriteBackground.html。

5．创建一个具有用来注册用户专业会议的表单的文档。在用户提交注册表单时，保存包含了用户信息（如姓名、公司等）的 cookie。若用户试图以相同的姓名第二次注册时，显示一个确认对话框询问用户是否想重新注册。在 Data Disk 的 Tutorial.08 文件夹内将文档存为 ConferenceRegistration.html。

6．创建一个具有预定客房表单的文档。在用户预定客房时，保存包含了用户路线的 Cookies。还请创建多个按钮用来在警告对话框中重新显示用户路线。将 Cookies 设置为访问一天后过期。在 Data Disk 的 Tutorial.08 文件夹内将文档存为 HotelReservations .html。

8.2 安　　全

本节目标

在本节将学习：

- ✧ JavaScript 安全所关心的内容
- ✧ 同源策略
- ✧ 署名脚本和数字证书
- ✧ 怎样创建测试证书
- ✧ 怎样使用特权
- ✧ 怎样签署 JavaScript 程序
- ✧ 怎样启动代码库委托

8.2.1　JavaScript 安全所关心的内容

现在听到有关 Internet 安全漏洞的消息很普遍。企业和政府网址被未验证的用户侵入看起来十分常见，信用卡号和其他私有信息在 Internet 传输途中经常被窃取。为了与安全的不法行为作斗争，已经开发出了防火墙等技术，它们将硬件和软件结合起来禁止访问连接到 Internet 上的私有网络。另外，为了保护信息，Netscape 开发了安全套接字层协议来加密数据并在安全的连接上进行传输。这些类型的安全技术在 Internet 领域内很有效。然而，JavaScript 程序被下载在本地执行，即客户计算机的浏览器内，并不受安全技术（如防火墙和安全套接字层）的控制。

提示：Internet 安全是一个大的不断发展的问题。本教程讨论安全仅因为它与 Web 浏览器和客户端 JavaScript 相关。

Web 最初是只读的，因为它的主要目的是定位和显示文档，它们存储在 Web 其他地方。随着编程语言的发展，如 Java 和 JavaScript，除了静态内容，Web 页现在还能够包括程序。在 Web 页内执行程序的能力导致了数种安全忧患。JavaScript 程序员最担心的安全领域是：

- ✧ 保护 Web 页和 JavaScript 程序不被恶意篡改。
- ✧ 个人客户信息的保密。
- ✧ 保护客户本地文件系统或网址不被窃取或篡改。

由于 HTML 和 JavaScript 程序代码的开放特性，任何人都能够通过选中 Netscape View 菜单内的 Page Source 或 Internet Explorer 察看菜单的源文件来读取 JavaScript 代码。能够使用 JavaScript 源文件来隐藏代码，尽管客户还能够察看源文件。例如，考虑下面的<SCRIPT>标签，它载入 HiddenScript.js 源文件：

```
<SCRIPT LANGUAGE="JavaScript1.2"
SRC="http://www.dongosselin.com/javascript/HiddenScript.js">
```

前面的<SCRIPT>标签能够被内嵌在 HTML 文档内，防止客户直接看到 HiddenScript.js 文件内的 JavaScript 代码。然而，任何人都能够在 Web 浏览器内打开它的 URL 和选中察看源文件来察看 HiddenScript.js 的内容。没有方法来隐藏 JavaScript 代码。潜在的窃取工作成果和脑力财富的行为是个大的忧患。然而，在不能控制对代码的访问时将会导致大的安全问题。在某个人能够访问您的代码时，不道德的人可能修改它并且利用它来窃取信息或对客户系统造成破坏。有两种安全特性（同源策略和脚本的数字签名）被用来保证代码不会被篡改。除了提供了安全特性，数字签署的脚本还验证脚本编写人的可信度，它使用数字证书技术。

另一个安全忧虑是 Web 浏览器窗口内个人客户信息的隐私性。您的 E-mail 地址、书签和历史记录都是有价值的信息，许多市场人员喜欢获取这些信息以便根据偏爱和厌恶进行

广告轰炸。JavaScript 程序没有安全缺陷，能够从您的 Web 浏览器中读取信息。为了保护隐私，JavaScript 包括了特权安全特性，它确定了什么类型的浏览器信息能够被脚本访问。使用特权和签名的脚本来指定脚本能够访问的信息类型。

Internet Explorer 不直接支持 JavaScript 程序的数字签名，也不支持 JavaScript 代码的特权维护。因为 Internet Explorer 仅支持某些 JavaScript 安全特性，如同源策略。本节主要是针对 Navigator 来讲的。

JavaScript 最重要的安全特性之一是它缺乏某些类型的功能。例如，许多编程语言包括了工程和方法让您读取、写入和删除文件。为了防止恶意的脚本窃取信息或由于改变或删除文件而造成的损坏，JavaScript 不允许一丁点文件维护。同样，JavaScript 不包括用来创建网络连接的机制。这种限制防止 JavaScript 程序渗入私用网络或内部网，从中窃取或破坏信息。JavaScript 也不能运行系统命令或执行客户上的程序。读取和写 cookie 是 JavaScript 仅有的一种能够访问客户的能力。然而，Web 浏览器严格控制 Cookies 并且不让从创建它们的域的外面访问 Cookies。

提示：可能听到的另一个 JavaScript 安全特性是数据污染，它将某些特定的属性标为私有或安全。Navigator 3 支持数据污染，但是 Navigator 4 就不支持了。因此本课程讲授的是 JavaScript 1.2，仅在 Navigator 4 和更高版本以及 Internet Explorer 中才支持它，所以这里不讨论数据污染。

8.2.2 同源策略

同源策略限制了一个窗口或帧内的 JavaScript 代码访问客户计算机上另一个窗口或帧内的 Web 页。为了让窗口和帧浏览和修改在另一个窗口和帧内显示的文档的重要属性，它们必须使用相同的协议（如 HTTP）并且在同一台服务器上。例如，来自下面两个域内的文档不能访问彼此的属性，因为它们使用不同的协议。第一个域的协议是 HTTP 而第二个域内的协议是 HTTPS（它在安全网络上使用）。

```
http://www.gosselin.com
https://www.gosselin.com
```

同源策略不仅应用于域名，而且还应用于文档所在的服务器。因此，来自下面两个域的文档不能访问彼此的属性，因为它们位于不同的服务器上，尽管它们在同一个域名 gosselin.com 下。

```
http://www.programming.gosselin.com
https://www.writing.gosselin.com
```

同源策略防止恶意的脚本修改其他窗口和帧内的内容，还防止了窃取个人浏览器信息

和安全 Web 页面内所显示内容。表 8-3 显示了支持同源策略的 JavaScript 对象和属性。

表 8-3　支持同源策略的 JavaScript 对象和属性

对　象	属　性
Document	读写限制：anchors、applets、cookie、domain、elements、embeds、forms、lastModified、length、links、referrer、title、URL 和表单名 仅限制写：所有其他属性
Image	src、lowsrc
Layer	src
Location	all
Window	find

提示：可以利用签署脚本和特权（在本节的后面部分讨论）来绕过同源策略限制。

作为没有同源策略将会发生什么情况的实例，请考虑一下 Document 对象的 src 属性，它确定了在窗口或帧内所显示的 URL。若客户在它的系统上打开了多个窗口或帧并且不存在同源策略，那么一个窗口或帧内的 Web 页面能够修改在另一个窗口或帧内显示的 Web 页面。有许多不道德或恶意的广告人员，他们强迫您仅浏览他们的 Web 页面。如果没有同源策略，私有网络和内部网也将处于危险之中。请考虑下面的情况：一个用户在一个 Web 浏览器中打开了 Internet 上一个页面，同时他还打开了来自他或她私有网络或内部网内的一个安全页面。在没有同源策略的情况下，Internet Web 页面将能够访问显示在私有 Web 页面上的内容。

同源策略还保证了 Web 页面设计的完整性。例如，没有同源策略，在一个窗口或帧内的一个帧将能够修改在另一个窗口和帧内的 JavaScript 对象的属性和 HTML 代码。为了理解同源策略是怎样防止发生这种类型的情况的，我们将创建一个帧集，其中一个帧使用 JavaScript 代码试图修改 Yahoo!主页的背景颜色，JavaScript 代码使用了 Document 对象的 bgColor 属性。对读取来自不同源的文档的 Document 对象的 bgColor 不存在缺陷。然而，改变 bgColor 属性受同源策略的控制。因为文档不是 Yahoo!域内的一部分，所以在试图执行此代码时将会收到错误消息。

创建一个帧集，其中一个帧使用 JavaScript 代码试图修改 Yahoo!主页的背景颜色，JavaScript 代码使用了 Document 对象的 bgColor 属性：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．输入下面创建一个具有两个帧的帧集的行。代码让 Yahoo!主页显示在第二个帧内。

```
<HTML>
<FRAMESET COLS="20%,*">
    <FRAME NAME="wrongframe" SRC="WrongOrigin.html">
    <FRAME NAME="yahooframe"
        SRC="http://www.yahoo.com/">
</FRAMESET>
```

```
</HTML>
```

3．在 Data Disk 的 Tutorial.08 文件夹内将文件存为 MainFrame.html。

4．下一步，在文本编辑器或 HTML 编辑器内创建另一个新文档。

5．添加下面一个简单的表单，它包含了一个简单的 Change Color 按钮。按钮使用了 onClick 事件来试图修改包含了 Yahoo!主页的帧的背景颜色。

```
<HTML>
<FORM>
<INPUT TYPE="button" VALUE=" Change Color "
      onClick="parent.yahooframe.document.bgColor =
            'yellow';">
</FORM>
</HTML>
```

6．将文件存为 WrongOrigin.html，接着关闭它。在浏览器内打开 MainFrame.html。在载入 Yahoo!之后，单击左帧内的“ Change Color ”按钮。应该会收到一条错误消息。图 8-5 显示了 Navigator 生成的错误消息。

帮助：若使用的是 Internet Explorer 5 或高于 4.0 的 Navigator，那么为了查看错误消息，需要在 Internet Explorer 的地址框中或 Navigator 的位置框中输入 javascript:。

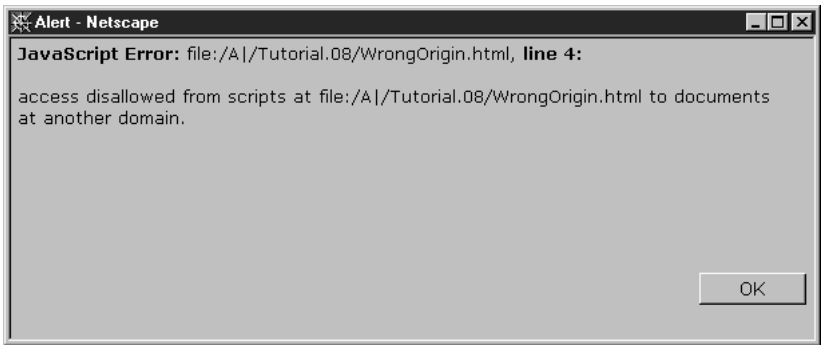


图 8-5　Navigator 中错误源的错误消息

7．关闭 Web 浏览器。

存在多种情况，其中想让不同服务器上两个相关网址的两个文档能够访问彼此的属性。请考虑下面的情况，来自 programming.gosselin.com 的一个文档和来自 writing.gosselin.com 的一个文档被用在同一个帧内，并且其中的一个文档被指定来更新另一个文档的属性。为了让来自同一域内不同源的文档访问彼此的属性，使用 Document 对象的 domain 属性。Document 对象的 domain 属性使用 document.domain ="域";将文档的源变为它的根域名。将语句 document.domain ="gosselin.com"; 添加到来自 programming.gosselin.com 和

writing.gosselin.com 的文档中，它让文档能够访问彼此的属性，尽管它们位于不同的服务器上。

提示：能够将 domain 属性设置为文档源的一个域，而不必将它设置为源外任何域。

8.2.3　签署脚本和数字证书

JavaScript 代码实际上是在 Web 浏览器内创建了一个小型程序，任何熟悉 JavaScript 语法的人都能编写它。不像购买的装在 shrink-wrapped 盒内软件程序，无法知道是谁编写的 JavaScript 程序，也无法知道它是否通过某种方式进行了修改（非作者所期望的）。篡改 JavaScript 尤其可能，因为任何人都能够浏览、拷贝和修改一个 JavaScript 程序。在计算机上运行不是有目的安装的程序具有两个危险。首先，不具有机会来决定让程序做想让它做的事情。第二，不知道是否能够信任程序的作者。当在计算机上安装新程序时，如字处理程序，通常能够看到一些安装选项。它询问安装程序的位置，它是否是默认的字处理程序等。若是众所周知的公司，如 Microsoft 或 Netscape 开发的程序，能够十分肯定它是可信任的和安全的（只要是从著名的代理商那里购买的装在 shrink-wrapped 盒内的软件）。

尽管 JavaScript 程序不像传统的软件那样需要安装，但是具有严格的规则控制使用 JavaScript 的程序访问 Web 浏览和计算机系统。例如，若没有授予 JavaScript 程序特权，那么 JavaScript 程序就不能关闭 Web 浏览器或访问您的信息。特权是指一种许可，它被授予以便访问受限制的特性或 JavaScript 程序通常不能访问的信息。对于一个 JavaScript 程序访问特权，它必须向具有授予或拒绝请求权利的用户提出请求。通过此种安全模型，决定授予 JavaScript 程序多少安全访问取决于客户。考虑一下，在试图将表单提交给 E-mail 地址时将会发生什么情况。当试图邮寄一个表单时，将会看到一个对话框，它询问是否想给 JavaScript 程序授予 E-mail 特权。其他类型的特权还包括历史列表和访问您的偏爱。

提示：JavaScript 操作的规则就是众所周知的“沙箱”。沙箱术语来源于 Java 编程，它定义了一个谨慎控制的环境，编程语言在其中能够操作。

通常， JavaScript 程序禁止操作特权。然而，有时客户需要授予许可使用特权，例如在邮寄表单时。在 JavaScript 程序请求特权的情况下，若不知道是谁编写的程序，那么授予许可将是危险的。为了识别 JavaScript 程序的编写者，Navigator 支持脚本数字签名。数字签署清楚地标识了 JavaScript 的编写者并且保证 JavaScript 的源格式没有被修改过。数字证书是一个 JavaScript 程序的编写者配置在签名脚本上的电子身份。由已知的、可以信赖的组织如证书权威机构（CA）来颁发数字证书。证书权威机构核查数字证书所有者或委托人的身份。委托人或实体是指数字证书的所有者。一些知名的证书权威机构包括 VeriSign、Equifax 和 GlobalSign。

提示：尽管数字证书并不足够确保能够信任它的所有者，若证书权威机构已经核查了

所有者的身份，那么这是安全的。

Security Info 对话框显示了与数字证书相关的信息以及用于 Navigator 安装的其他安全信息。为了使用 Security Info 对话框，请选择 Navigator 中 Window 菜单下的 Security Info。Certificates 菜单让您管理安装 Navigator 时注册的证书，包括您的个人证书和 Web 网址证书。还能够看到 Netscape 能够识别的证书权威机构。图 8-6 显示了选中 Certificates 菜单上的 Yours 选项后所打开的 Security Info 对话框的内容。

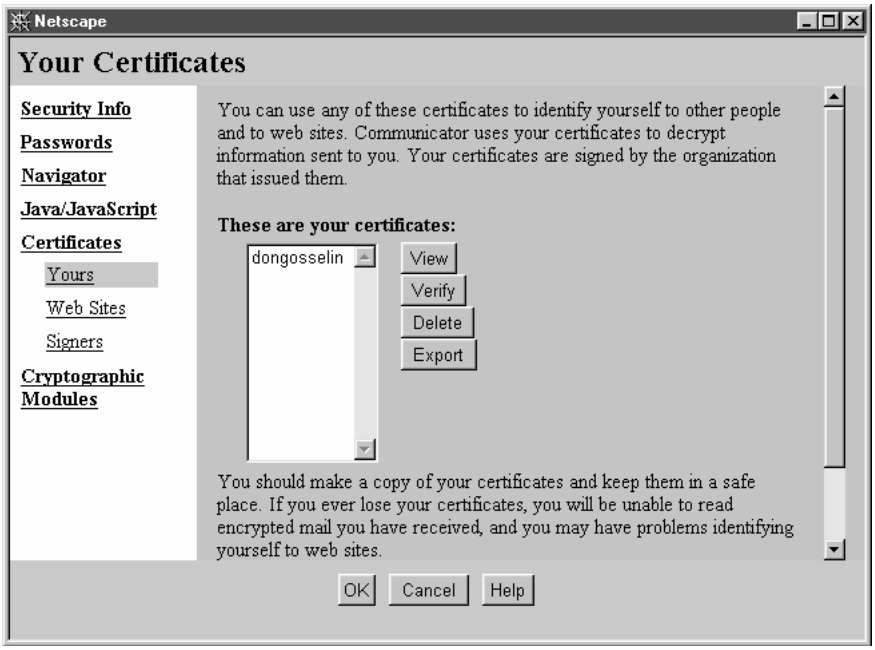


图 8-6　Sercurity Info 对话框的 Certificates 菜单。

提示：若滚动到 Certificates 菜单上的 Yours 选项的底部，将会看到一个 Get a Certificate 按钮，它将您指引到能够帮助获取自己数字证书的网址。还能够使用 Import a Certificate 按钮导入已有的证书。

下面是签署脚本所需的步骤：

1．取得一个数字证书或创建一个测试证书。

2．编写 JavaScript 程序并在其中包括所有特权请求。

3．准备签署 JavaScript 程序。

4．使用 Netscape 的 SignTool 程序签署 JavaScript 程序。

本节仅适用于 Navigator，因为 Internet Explorer 不支持 JavaScript 代码的数字签名。为了执行本节内的练习，需要 Navigator 和 Netscape 的 Signing Tool（或 Signtool）程序拷贝。Signtool 是 Netscape 开发的一个 Perl 程序，它是创建签署脚本所必需的。能够从 http://developer.netscape.com/software/signedobj 下载 Signtool。已有用于不同操作系统的各

种版本的 Signtool。下载的 Signtool 程序已被压缩成 ZIP 文件。在下载完 ZIP 文件之后，请将它存储在想安装 Signtool 程序的目录下，并使用 PKUNZIP 或任何其他能够处理 ZIP 文件的压缩/解压缩应用程序进行解压缩。解压缩 ZIP 文件是安装 Signtool 程序所需的惟一一个步骤。本节内的步骤解释了怎样使用 Windows 操作系统下的 Signtool。为了方便在 Windows 操作系统的命令提示符下运行 Signtool 程序，可以将 Signtool 所安装的目录添加到 PATH 语句中。若使用另一种类型的操作系统，如 Macintosh 或 UNIX，那么请参阅 Signtool 在线文档。

提示：PATH 语句告知操作系统在命令提示符下运行命令时，在哪里搜索文件。不同平台设置 PATH 语句的方法不同。例如，在 Windows NT 下，将 PATH 语句存储在 AUTOEXEC.NT 文件中告知操作系统在哪里搜索文件。AUTOEXEC.NT 文件中类似于 PATH=c:\dos;c:\winnt 的 PATH 语句指引 Windows NT 每次在命令提示符下执行命令时，在 c:\dos 和 c:\winnt 文件夹内搜索文件。

提示：本节只包括了使用 Signtool 的基本知识。Signtool 的完整文档可以从 http://developer.netscape.com/docs/manuals/signedobj/signtool/index.htm 下载。

创建测试证书

为了学习怎样使用数字证书，将使用 Signtool 来创建一个测试证书。若没有 Signtool 创建的测试证书，为了执行本节内的练习，那么将需要从证书权威机构购买一个真正的证书。创建的测试证书仅用于测试目的，并不能作为证书权威机构颁发的、公认的证书。若想合法地签署 JavaScript 程序，将需要购买自己的证书。

在Windows操作系统下，Signtool是一个命令行程序。创建测试证书的命令行语法是 signtool -G certificate name -d certificates directory。在创建测试证书之前，必须退出 Navigator，否则可能破坏证书数据库。若在certificate name中包括了空格，那么必须将名称用引号括起来。同样，若在certificates directory中包括了空格，那么它的名称也必须用引号括起来。certificates directory是证书数据库所在的位置。在Windows 95/98系统下，certificates directory通常是：c:\Program Files\Netscape\Users\default\。对Windows NT系统，certificates directory默认的位置是c:\Program Files\Netscape\Navigator\Users\user name\。在Unix系统下，目录是~/.netscape。若定位certificates directory存在问题，那么请搜索包含了Cert7.db和Key3.db文件的目录。下面的代码在Windows 98操作系统下创建了一个dongosselin证书。

```
signtool -G dongosselin -d "c:\Program
Files \Netscape \Users \default \"
```

提示：前面的实例被分为两行是因为有限的空间。在输入此命令时，按下回车键之前，请保证在单行上输入整条命令。

Signtool 的参数区分大小写。因此，在前面的代码中-G（用于生成）必须以大写字符输入，并且-d 参数（用于目录）必须以小写字符输入。

Navigator 中的证书通过口令来保护。为了创建一个测试证书，必须首先在安装 Navigator 时指定一个证书口令。

下一步，将创建一个证书口令。若已经输入了一个证书口令，那么请跳过下面的步骤。

创建证书口令：

1．若需要，请启动 Navigator。

2．从 Window 菜单下选中 Security Info 来显示 Security Info 窗口，接着选中 Passwords 菜单。

3．单击“ Change Password ”或是“ Set Password ”按钮（取决于 Navigator 的版本）来显示 Setting Up Your Navigator Password 对话框。在 Password 文本框内输入口令，接着在 Type it again to confirm 框内再次输入此口令。请记下此口令因为将需要它来创建测试证书。

4．单击“ OK ”关闭 Setting Up Your Navigator Password 对话框。在 Password 选项对话框的底部的单选按钮组内，请确保选中了第一个单选按钮。第一个单选按钮在首次需要您的证书时提示输入 Navigator 口令。

5．单击“ OK ”关闭 Security Info 窗口。

6．关闭所有的 Navigator 实例。请记住，若在 Navigator 正在运行时创建测试证书，那么可能会破坏证书数据库。

下一步，将使用 Signtool 创建一个测试证书：

1．从 Windows 开始菜单选中运行，接着输入 command（Windows 95/98 下）或 cmd（Windows NT 下），并按下回车键，打开一个命令行窗口。

2．若 Signtool 所在的目录不是系统路径的一部分，那么请切换到安装 Signtool 的目录下。

3．输入 signtool -G your name -d certificate directory 并按下回车键。若在 your name 中包括了空格，或是在 certificate directory 中存在空格，那么请用引号将它们括起来。还请记住 -G 和 -d 参数区分大小写。例如，若 certificate directory 是 c:\Program Files\Netscape\Users\default，为了创建 dongosselin 证书，输入 signtool –G dongosselin –d "c:\Program Files\Netscape\Users\default"。

4．Signtool 将警告在运行此命令之前，必须关闭 Communicator，接着提示输入 y 继续或是其他任意键取消。若确定不存在任何正在运行的 Communicator 实例，那么请输入 y。接着 Signtool 向您提示其他几种类型的可选信息，如证书的通用名、组织、州或省等。必须输入的惟一选项是证书数据库的口令。在输入证书数据库口令之后，Signtool 将在屏幕上显示几行，包括了一行说明 certificate “ certificationname ”added to database。

5．输入 exit 并按下回车键关闭命令行窗口。

6．为了核查证书，请启动 Netscape。从 Window 菜单选中 Security Info 来显示 Security Info 窗口。在 Security Info 窗口内单击 Yours 菜单，单击您的证书名，接着单击“ View ”按钮来显示一个说明了您的证书相关信息的窗口。图 8-7 显示了测试证书的实例。

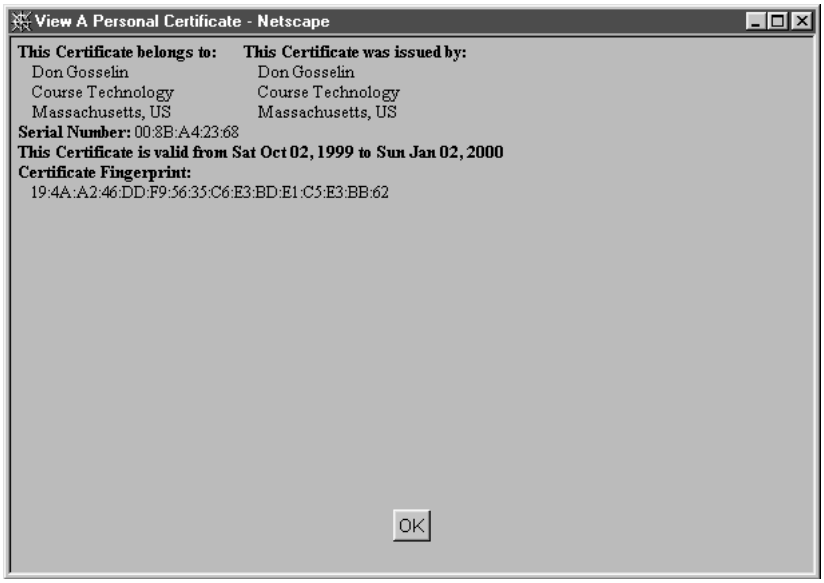


图 8-7　测试证书

提示：在配置签署脚本来请求特权之前，将不能浏览“官方”版本的证书（像您在浏览中所看到的那样）。

使用特权

为了在 JavaScript 使用受限的特权，必须访问它的目标。目标是一个由相关的特权类型组成的类别。目标名以单词 Universal 开始，例如，为了在 JavaScript 内发送邮件，请求访问 UniversalSendMail 目标。用于 JavaScript 请求特权的目标是：

- ✧ UniversalBrowserRead
- ✧ UniversalBrowserWrite
- ✧ UniversalBrowserAccess
- ✧ UniversalFileRead
- ✧ UniversalPreferencesRead
- ✧ UniversalPreferencesWrite
- ✧ UniversalSendMail

表 8-4 列出了每种目标可用的特权。

JavaScript 使用一个 netscape.security.PrivilegeManager Java 类来请求特权。netscape.security.PrivilegeManager 类包含了两个方法：enablePrivilege()和 revertPrivilege()。enablePrivilege()方法使用对话框请求客户的一个特权。客户具有授予或拒绝特权请求的权利。enablePrivilege()方法只接受一个用引号括起来的参数，它包含了目标名。若授予了特权，可以使用 revertPrivilege()方法来撤回特权。第一条语句使用 enablePrivilege()方法授予访问 UniversalSendMail 目标的权限。第二条语句使用 revertPrivilege ()方法撤回访问 UniversalSendMail 目标的权限。

表 8-4 目标及其相关特权

属 性	描 述
UniversalBrowserRead	检索 History 对象的 any 属性的值 检索 DragDrop 事件的 data 属性的值 使用 about:URL 而不是使用 about:blank
UniversalBrowserWrite	使用 Window 对象的 enabelExternalCapture()方法来捕获从不同服务器上载入的页面内的事件 使用 Window 对象的 moveBy()或 moveTo()方法将窗口移出屏幕外 读取或设置 Event 对象的 any 属性 读取或设置 Window 对象的所有下列属性：locationbar、menubar、personalbar、scrollbars、statusbar 和 toolbar 使用 Window 对象的 close()方法无条件地关闭浏览器窗口 使用 Window 对象的 innerWidth()或 innerHeight()方法将窗口的内部宽度和高度设置为比 100×100 小或比能够容纳它的屏幕大的尺寸 使用 Window 对象的 open()方法来创建一个不具有标题栏的窗口 使用 Window 对象的 open()方法将窗口放置在屏幕外 使用 Window 对象的 open()方法来使用 alwaysLowered、alwaysRaised 或 z-lock 使用 Window 对象的 open()方法创建一个比 100×100 小或比能够容纳它的屏幕大的窗口 使用 Window 对象的 resizeTo()或 resizeBy()方法将窗口的尺寸调整为比 100×100 小或比能够容纳它的屏幕大的尺寸
UniversalFileRead	在本地读取任何文件
UniversalPreferencesRead	使用 Navigator 对象的 preferences()方法读取所有引用
UniversalPreferencesWrite	使用 Navigator 对象的 preferences()方法写入所有引用
UniversalSendMail	将表单提交给 mailto:或 news:URL

```
netscape.security.PrivilegeManager.
        enablePrivilege("UniversalSendMail");
netscape.security.PrivilegeManager.
        revertPrivilege("UniversalSendMail");
```

在前面的实例中，若用户授予对 UniversalSendMail 特权的请求，那么 JavaScript 程序就能够提交一条邮件消息。若用户拒绝请求，那么将显示一条 Navigator 错误消息。注意客户基于每个事例来评估每个特权请求。在此种方式中，客户可以为每个委托人创建他自己的安全数据库。若客户在特权请求对话框内未选中 Remember this decision 复选框，或是使用 revertPrivilege()方法撤回了特权，那么特权将仅维持当前浏览器会话一段时间。若用户选中了 Remember this decision 复选框，那么对以后的任何请求特权都将自动被授予那个特

殊的委托人。

应该仅使用完成任务所需的最小的目标类型。例如，若仅需读取浏览器的设置，那么使用 UniversalBrowserRead 目标而不是 UniversalBrowserWrite 目标。若使用 UniversalBrowserWrite 目标，那么将为使用您程序篡改用户系统的人开了一个后门。使用只读的 UniversalBrowserRead 减少了这种风险。对所有目标类型，尤其是对那些能够真正对用户系统进行写操作的类型。在使用完它们之后立即撤回特权是一个不错的习惯。在使用完特权之后立即撤回，还有助于减少别人使用您的程序窃取信息或篡改用户系统的风险。

提示：若用户为一个特殊的特权选中了 Remember this decision 复选框，那么就只有用户能够撤回该特权。用户能够使用 Security Info 窗口中的 Java/JavaScript 菜单编辑委托人的特权列表来撤回特权。

下一步，将开始创建在浏览中所看到的试图将用户的主页重定向为 Course Technology 主页的程序。为了创建此程序，将使用 UniversalPreferencesRead 和 UniversalPreferencesWrite 目标。通过这些目标能够访问 Navigator 首选设置。为了改变用户的主页，将修改 browerser.startup.homepage 首选设置。

提示：在 Navigator 中能够设置许多种类型的首选。为了取得完整的 Navigator 首选列表，请访问 http://developer.netscape.com/docs/manuals/deploymt/jsprefs.htm。

开始创建改变用户主页的程序：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．输入文档的起始的<HTML>、<HEAD>和<SCRIPT>段。

```
<HTML>
<HEAD>
<TITLE>Change Home Page</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

3．添加下面的函数，它读取用户的主页设置（只要授予了 UniversalPreferencesRead 目标许可），接着向用户显示一个确认对话框，询问他们是否想将他们的主页改为 Course Technology 的主页。若用户单击“ OK ”，那么将调用 writeHomePage()函数。

```
function readHomePage() {
    netscape.security.PrivilegeManager.enablePrivilege(
        "UniversalPreferencesRead");
    var curHomePage =
        navigator.preference("browser.startup.homepage");
    var changePage = confirm (
        "Do you want to change your personal home page
```

```
from "+ curHomePage +
" to Course Technology's home page?");
        if (changePage == true)
                writeHomePage();
}
```

4．下一步输入将用户主页改为 Course Technology 主页的 writeHomePage()函数，若授予了 UniversalPreferencesWrite 目标许可。

```
function writeHomePage() {
        netscape.security.PrivilegeManager.enablePrivilege(
                "UniversalPreferencesWrite");
        navigator.preference("browser.startup.homepage",
                "http://www.course.com")
        alert(
                "Your home page has been changed to Course
Technology.");
}
```

5．输入结束的<HEAD>和<SCRIPT>标签以及起始的<BODY>标签。

```
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
<BODY>
```

6.输入下面的表单段,它只包含了一个使用 onClick 事件处理器来调用 readHomePage()函数的按钮。

```
<FORM>
<INPUT TYPE="button"
                VALUE=" Change Home Page to Course Technology "
        onClick="readHomePage();">
</FORM>
```

7．添加下面的代码来结束<BODY>和<HTML>标签。

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

8．为了保存文件，请在 Data　Disk 的 Tutorial.08 目录内创建一个 homepage 文件夹。

接着在 Data Disk 上新创建的目录内将文件存为 HomePage.html。在能够测试程序之前，必须使用您的数字证书签署了它。

签署 JavaScript 程序

签署程序将会创建一个 JAR 文件，其中包含了此程序 JavaScirpt 代码的数字签名。JAR 文件是一种特殊的压缩文件，它的后缀为.jar 并且包含了签署过的脚本信息。签署一个 JavaScript 程序并不会修改源 HTML 或 JavaScript。相反，签署过的 JavaScript 代码以及委托人的证书信息被存储在 JAR 文件中。源 HTML 和 JavaScript 代码以及 JAR 文件一起被配置在 Web 服务器上。在签署过的程序从客户请求特权信息时，Navigator 确定程序的 JAR 文件的位置并且将签署过的 JavaScript 代码和 HTML 及 JavaScript 源文件中的 JavaScript 代码进行比较。若它们不匹配，那么 Navigator 将忽略特权请求，因为代码可能已经被篡改或破坏了。若代码和签名匹配，那么 Navigator 将在特权请求对话框内显示委托人的证书信息。

提示：能够从 Navigator 的 View 菜单选中 Page Info 在 Web 页请求特权之前查看它的数字证书。

签署 JavaScript 程序由两步组成：准备签署程序和运行 Signtool 程序（它创建 JAR 文件）。使用<SCRIPT>标签的 ARCHIVE 和 ID 属性准备签署的程序。ARCHIVE 属性为签署过的脚本 JAR 文件指定所想使用的名称。必须在文档的第一个<SCRIPT>标签内包括 ARCHIVE 属性，并且只能包括一次。ID 属性是赋给每段 JavaScript 代码段的惟一标识。必须为签署过的脚本内的每段 JavaScript 代码段赋给一个惟一的 ID 标识，否则 Navigator 将整个脚本视为未被赋值。包含在 HTML 标签内的事件处理器和包含在<SCRIPT>…</SCRIPT>标签对内的代码是所有的 JavaScript 代码段，并且必须包括 ID 属性。图 8-8 显示了一个准备签署的程序。请注意包含了 JavaScript 代码源文件的<SCRIPT>…</SCRIPT>标签、内嵌的<SCRIPT>…</SCRIPT>段以及事件处理器都具有 ID 属性。

若想签署未包括<SCRIPT>…</SCRIPT>标签对的文档，那么必须创建一个空的<SCRIPT>…</SCRIPT>标签对来容纳 ARCHIVE 属性。与 ID 属性不一样，ARCHIVE 属性仅在<SCRIPT>标签内有效。下面的代码仅包含了一个显示警告对话框的按钮。在按钮前的空<SCRIPT>…</SCRIPT>标签对对被签署的文档是必需的。

提示：请记住 ARCHIVE 属性仅需出现一次，必须被放在文档内的第一个<SCRIPT>标签内，并且仅在<SCRIPT>标签内有效。

下一步将准备签署的 HomePage.html 文件：

1．返回到文本编辑器或 HTML 编辑器内的 HomePage.html 文件。

2．将属性 ARCHIVE="homepage.jar"添加给头段中的<SCRIPT>标签为签署过的脚本指定 JAR 文件。还添加属性 ID="s1"为此标签创建一个惟一的 ID。

3．为表单内的<INPUT>标签添加名为 b1 的 ID 属性。

4．保存并关闭 HomePage.html 文件。

```
<HTML>
<HEAD><TITLE>Signing</TITLE>
<SCRIPT ARCHIVE="SignedFile.jar"
        SRC="SourceFile.js" ID="s1">
</SCRIPT>
<SCRIPT ID="s2">
function hideElements() {
        netscape.security.PrivilegeManager.
                enablePrivilege("UniversalPreferencesWrite");
        navigator.preference(
                "browser.background_color", "blue");
}
</SCRIPT>
</HEAD>
<BODY>
<FORM>
<INPUT TYPE="button" VALUE=" Hide Screen Elements "
        ID="bt1" onClick="hideElements();">
</FORM>
</BODY>
</HTML>
```

图 8-8　准备签署的 JavaScript 程序

既然 HomePage.html 文件准备用于签署，那么可以使用 Signtool 签署它。使用 signtool -J -k certificate_name files_path -d certificates_path 来签署 JavaScript 程序。-J 参数表明正在签署的 JavaScript 文件，-k 参数指出了所使用的证书，-d 指出了证书目录的位置。还包括了证书名、想签署文件的路径以及证书目录的路径。请注意 Signtool 以数字形式签署指定目录内的所有文件。

签署 HomePage.html 文件：

1．从 Windows 开始菜单选择运行，接着输入 command（Windows 95/98）或 cmd（Windows NT），并按下回车键。将会打开一个命令行窗口。

2．切换至 Signtool 所安装的目录，若包含 Signtool 的目录是系统目录的一部分，那么可以跳到下一步。

3．输入 signtool -J -k certificate name -d "certificates directory " "files directory"，用在本节前面所创建的证书名代替 certificate name。还请用包含 HomePage.html 文件的目录代替 files directory。例如，为了签署 A:驱动器内的 Data Disk 的 Tutorial.08 文件夹内 homepage 文件夹内的 HomePage.html 文件，将输入 signtool –J –k dongosselin –d "c:\Program Files\Netscape\Users\default" a:\Tutorial.08\homepage。若目录中包含了空格，那么请确保用引号将目录名括起来。Signtool 将开始生成署名，接着提示输入证书口令。请输入口令并按下回车键。最后一行将说明包含文件的目录已被成功签署。

4．输入 exit 并按下回车键来关闭命令行窗口。

5．从 Data Disk 的 Tutorial.08 文件夹在 Navigator 内打开 HomePage.html。（请记住，签署过的文件名必须与 JAR 文件在相同的位置内）单击“Change Home Page to Course Technology”按钮。应该收到一条提示，它请求读取首选的额外特权。若授予此特权，将收到一个警告对话框询问是否想将您的主页改变为 Course Technology 主页。单击“OK”按钮显示另一条提示，它请求修改您首选的额外特权。授予额外特权改变您的个人主页。对每条特权请求，若拒绝请求，那么将会收到一条 Netscape 错误消息并且代码将会终止。在授予或拒绝特权之前能够单击“Certificate”按钮来查看您的数字证书。

在签署 JavaScript 程序之后，请为服务器配置 HTML 文件、JavaScript 源文件和 JAR 文件。需要记住的十分重要的一点是若修改了签署过的程序内的任何代码，那么必须重新运行 Signtool。既然 Navigator 将每段代码与 JAR 文件内的电子签名进行比较，那么即使细微的修改都将导致 Navigator 认为程序未签署。

开启代码库委托

随着开发将被数字签署的并且包括了特权请求的 JavaScript 程序，将会发现每次细微修改都重新签署脚本是十分繁琐的。若作出了修改，但是不重新签署脚本，那么 Navigator 将忽略任何特权请求。为了使测试和开发数字签署过的脚本更简单，能够开启代码库委托。代码库委托将 URL 的源（如 Internet 域或本地文件系统）认可为委托人，并且将数字证书所有者认可为委托人。这种认可让 JavaScript 程序在未被签署的情况下请求特权。应该将代码库委托仅用于在系统上进行测试和开发目的。在对请求特权的程序的性能满意之后，在让它服务之前，必须数字签署它。若未签署包含特权的脚本，那么特权请求将被忽略——若用户未在他们的系统上开启代码库委托。

为在系统上开启代码库委托，应该关闭任何 Navigator 实例，接着在系统上 Netscape 文件夹内确定 JavaScript 首选文件 prefs.js 的位置。prefs.js 文件所在的目录取决于系统。对 Windows 95/98 操作系统，prefs.js 文件的位置通常是 c:\Program Files\Netscape\Users\default\。请在文本编辑器（如 Notepad）中打开 prefs.js，并在文件末尾添加下面的行：

```
user_pref("signed.applets.local_classes_have_30_powers",
        true);
```

提示：在开启代码库委托之前，请确信理解了 prefs.js 文件中所要编辑的行，否则可能会意外地对所安装的 Navigator 产生问题。还请记住开启代码库委托仅用于测试目的——永远记住在测试完数字签署文件之后立即禁止开启代码库委托。

若为 pref.js 添加了前面的行，并且保存、关闭了该文件，那么在下次启动 Navigator 时将会开启代码库委托。为了禁止开启代码库委托，需要重新打开 prefs.js 并且删除为文件添加的代码行。

8.2.4 总结

- ✧ 同源策略限制了客户计算机上一个窗口或帧内的 JavaScript 代码怎样访问另一个窗口或帧内的 Web 页面。
- ✧ 同源策略不仅用于域名，还用于文档所在的服务器。
- ✧ 同源策略防止恶意的脚本修改另一个窗口和帧内的内容，也防止窃取私有浏览器信息和在安全 Web 页面上所显示的信息。
- ✧ Document 对象的 domain 属性将文档的源改变为它的根域名，请使用 document.domain = "域"。
- ✧ 特权是指受限的特性或通常 JavaScript 程序不可用的信息。为了让 JavaScript 程序访问特权，必须向客户（他具有授予或拒绝请求的权利）请求特权。
- ✧ 数字签名清楚地标识了 JavaScript 程序的编写者，并且保证未对源 JavaScript 程序进行过任何修改。
- ✧ 数字证书是一个 JavaScript 程序的编写者为签署过的脚本所配置的电子身份。
- ✧ 可以信赖的、知名的证书权威机构（CA）发放数字证书。证书权威机构核查了数字证书所有者（或委托人）的身份。
- ✧ 委托人（或实体）是指数字证书的所有者。
- ✧ Security Info 对话框提供了数字证书的相关信息，还提供了其他安装 Navigator 时所配置的安全信息。
- ✧ Signtool 是 Netscape 开发的一个 Perl 程序，它对签署脚本是必需的。
- ✧ 目标是由一组相关特权类型组成的分类。目标名以 Universal 开头。
- ✧ enablePrivilege()方法使用对话框来请求客户的受限特权。
- ✧ 若授予了特权，那么可以使用 revertPrivilege()方法来撤回特权。
- ✧ 签署程序将会创建一个 JAR 文件，它包含了程序 JavaScript 代码的数字签名。
- ✧ 使用<SCRIPT>标签的 ARCHIVE 和 ID 属性来准备签署 JavaScript 程序。
- ✧ 代码库委托将 URL 源（如 Internet 域或本地文件系统）认可为一个委托人，也将数字证书的所有者认可为委托人。

8.2.5 问题

1．Windows 对象的哪种属性受同源策略的限制？

a．move 和 resize 属性

b．open 和 close 属性

c．find 属性

d．所有属性

2．Document 对象的哪种属性将文档的源改为它的根域名？

a．domain

b．root

c．server

d．source

3．怎样防止用户察看 JavaScript 代码？

a．数字签署代码

b．将它保存在 JavaScript 源文件中

c．使用 escape()方法对它编码

d．不能防止用户察看 JavaScript 代码

4．JavaScript 程序怎样才能取得特权的访问？

a．请求客户

b．与客户浏览器位于相同的服务器上

c．使用 Window 对象的 privilege()方法

d．JavaScript 程序自动具有特权的访问

5．数字签署清楚地标识了 JavaScript 程序编写者的身份并且____。

a．允许 JavaScript 程序将 DHTML 融入 Web 页面

b．每次客户打开 JavaScript 程序的 Web 页面时要求一个口令

c．自动提供对客户系统的安全特性的访问

d．保证未对 JavaScript 程序的源格式进行过修改

6．数字证书所有者被称为委托人或____。

a．签名人

b．实体

c．个人

d．证书拥有者

7．数字证书由____发放。

a．InterNIC

b．联邦政府

c．Software Publishers Association

d．证书权威机构

8．在 Windows 95/98 系统上使用 Signtool 生成测试证书的正确语法是____。

a．signtool -G dongosselin -d "c:\Program Files \Netscape \Users \default \"

b．signtool -g dongosselin -D "c:\Program Files \Netscape \Users \default \"

c．signtool -G dongosselin -d

d．signtool -D dongosselin -G "c:\Program Files \Netscape \Users \default"

9．在____对话框中可以察看本地所安装的证书的相关信息。

a．Java Console

b．Security Info

c．Preferences
d．Encoding
10．特权种类被认为是____。
a．类
b．分组
c．媒介物
d．目标
11．请求特权访问的正确语法是____。
a．enbalePrivilege("UniversalSendMail");
b．netscape.PrivilegeManager. enbalePrivilege("UniversalSendMail");
c．netscape.security.enbalePrivilege("UniversalSendMail");
d．netscape.security.PrivilegeManager.enbalePrivilege("UniversalSendMail");
12．使用____方法来撤回特权。
a．removePrivilege()
b．revertPrivilege()
c．revokePrivilege()
d．disablePrivilege()
13．在签署过程中将创建何种类型的文件？
a．JAR
b．ZIP
c．ENC
d．ARC
14．使用 Signtool 签署脚本的正确语法是____。
a．signtool -J certificate_name files_path –d certificates_path
b．signtool -J -k certificate_name d certificates_path
c．signtool -J -k certificate_name files_path
d．signtool -J -k certificate_name files_path -d certificates_path
15．____将 URL 源（如 Internet 域或本地文件系统）认可为一个委托人，也将数字证书的所有者认可为委托人。
a．Domain 名和委托人
b．未签名的可信赖委托人
c．同源委托人
d．代码库委托

8.2.6 练习

1．在 Internet 上搜索与 Web 页面数字安全相关的信息。请撰写一篇一页纸的论文略述

Web 页面安全的不同方法，并指出此技术发展的方向。

2．许多受欢迎的证书权威机构都在美国之外。请撰写数段文章解释为什么会如此？

3．创建一个签署过的文档，它能够根据您的偏爱来改变页面大小和移动页面，接着将浏览器重定向常用的主页。在 Data Disk 的 Tutorial.08 文件夹内将文档存为 ResizeWindow.html。

4．创建一个签署过的文档，它使用 window.open()方法和 alwaysRaised 参数创建一个新窗口。在 Data Disk 的 Tutorial.08 文件夹内将文档存为 AlwaysOnTop.html。

5．创建一个签署过的文档，它打开一个新窗口并显示当前窗口的浏览历史。在 Data Disk 的 Tutorial.08 文件夹内将文档存为 BrowsingHistory.html。

第 9 章　调试 JavaScript

案例

编写的 JavaScript 程序越多，碰到的错误消息也就越多。另外，程序从来不按照所想要的方式运行。既然现在了解了大部分 JavaScript 编程基础，并且 WebAdventure 让您不断地为他们的客户编写 JavaScript 程序，那么现在该是深入 JavaScript 调试技术的时间了。

9.1　基本调试技术

本节目标

在本节将学习：

- ✧ 调试概念
- ✧ 怎样解释错误消息
- ✧ 怎样使用 alert()方法跟踪错误
- ✧ 怎样使用 write()和 writeln()方法跟踪错误
- ✧ 怎样使用注释来定位错误
- ✧ 其他调试技术

9.1.1　了解调试

无论他的经验、知识和能力如何，任何程序员都曾经在他们的程序中犯过错误。本教程一开始就介绍了，调试是跟踪和解决程序中错误的行为。不管用什么编程语言，调试都是程序员的一项基本技能。在本教程中，将学习帮助跟踪和解决 JavaScript 程序中错误的技术和工具。然而需要记住调试方法可能随浏览器的每个版本而不同。本教程中学习的技术和工具在 Internet Explorer 4 和 Navigator 4 下都可运行。

提示：许多高级编程语言都包括众多异常处理特性，它允许程序在执行过程中产生错误时进行处理。尽管在将来的版本中希望包括此能力，但是 JavaScript 1.2 不具有异常处理能力。

在程序中可能出现三种错误：语法错误、逻辑错误和运行时错误。在输入解释器不能识别的代码时将产生语法错误。JavaScript 中的语法错误包括无效语句或输入不正确的语句，例如，漏掉方法的反圆括号。其他类型的语法错误包括拼写错误或误拼的单词，例如，若输入语句 document.writln("Hello World");，在运行此程序时将会遇到语法错误，因为 writeln()被误拼为 writln()。同样，语法 Document.writeln("Hello World");也将造成语法错误，因为 Document 对象输入有误，输入的是大写 D（记住大部分 JavaScript 对象，如 Document 对象，在语句中以小写字母的形式输入）。

提示：编译语言中的语法错误，如 C++，也称为编译错误，因为它们通常在编译时被发现。既然 JavaScript 是解释语言，在程序执行之前将发现不了语法错误。

若在程序执行时 JavaScript 解释器遇到问题，那么此问题被称为运行时错误。运行时错误与语法错误不同，因为它们并不表示 JavaScript 语言错误。相反，在解释器遇到不能处理的错误时将产生运行时错误。例如，语句 customFunction();调用自定义的 JavaScript 函数 customFunction()。它不是一个语法错误，因为在 JavaScript 中创建自定义函数是合法的（通常是必要的）。然而，若创建函数失败并且在程序执行时试图调用此函数，那么将导致运行时错误，因为解释器不能找到此函数。下面的代码演示了另外一个运行时错误实例。在此例中，writeln()方法试图打印 messageVar 变量的内容。因为 messageVar 变量没有被声明（假定在文档中的其他脚本段中没有声明它），所以导致运行时错误。

```
<SCRIPT LANGUAGE="JavaScript1.2">
document.writeln(messageVar);
</SCRIPT>
```

逻辑错误是程序设计中的问题，它妨碍程序按照所期望的那样运行。任何程序内的逻辑是按照正确的次序执行各种语句和过程，它能够生成所需的结果。例如，在洗熨时，通常是先洗、干燥、熨烫，接着是折叠起来。若洗熨的程序是先熨烫、折叠、干燥，接着是洗，那么将出现逻辑错误，并且程序不能正确执行。在计算机程序中的逻辑错误的一个实例是在想将两个数相除时却将它们相乘，如下面的代码所示：

```
<SCRIPT>
var divisionResult =10 *2;
document.write("Ten divided by two is equal to "
      +divisionResult);
</SCRIPT>
```

另一个逻辑错误的实例是死循环，其中循环语句永不结束，因为它的条件语句永不被更新或永不为假。下面的代码创建了一个 for 语句，它将导致死循环逻辑错误，因为 for 语句构造器的第三个参数从不改变计数变量的值。

```
for(var count =10;count >=0;count){
        alert("We have liftoff in "+count);
}
```

因为在前面实例中的 count 变量的值从不被更新，所以循环每次重复它的值都将继续为 10，导致不断显示包含了文本“ We have lefttoff in 10 ”警告对话框。为了更正此逻辑错误，请为 for 语句的构造器中的第三个参数添加递减运算符，如下：

```
for(var count =10;count >=0;--count){
        alert("We have liftoff in "+count);
}
```

9.1.2 错误消息

在解释器遇到语法或运行时错误时，定位 JavaScript 程序中错误的措施的第一行是所看到的错误消息。Navigator 和 Internet Explorer 在错误消息对话框中显示的两段重要的信息是错误在文档中的行号以及错误的描述。错误消息中的行号从 HTML 文档的起始处计起，而不是从脚本段起始计起。除了错误的行号和描述，Navigator 还显示出错代码片断。Internet Explorer 指出错误所在行中的字符并告诉是否是运行时错误。任何非运行时错误都将是语法错误。作为 Navigator 和 Internet Explorer 中显示的错误消息的实例，请考虑下面的函数，它将导致语法错误，因为它漏掉了反大括弧“ } ”。

```
function missingClosingBrace(){
        var message =
                "This function is missing a closing brace.";
        alert(message);
```

图 9-1 显示了前面实例导致的 Navigator 错误消息，图 9-2 显示了在 Internet Explorer 中的错误消息。

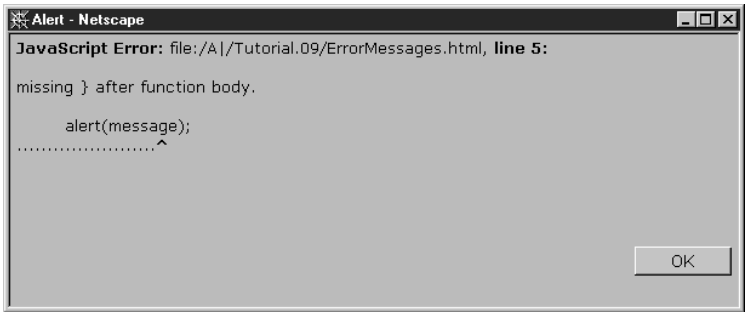


图 9-1 Navigator 错误消息

图 9-2　Internet Explorer 错误消息

帮助：若正在使用比 Internet Explorer 4 高的版本，那么接着需要从工具菜单中选择 Internet 选项并单击“高级”标签来启用错误通知。在高级标签上的浏览目录下，确保选中了显示每个脚本错误的通知复选框，接着单击“确定”按钮关闭对话框。若正在使用比 Navigator 4 高的版本，为了查看错误消息，不得不在 location 文本框中输入 javascript:并按下回车键。另外，在比版本 4 更高的浏览器中，看到的错误消息可能与在本教程图中所看到的错误消息样式不同。出现在本教程图中的错误消息是使用 Internet Explorer 4 和 Navigator 4 所看到的。

提示：注意 Navigator 错误消息指出了行 5，而 Internet Explorer 错误消息指出了行 4。Navigator 指出了大括弧应被放置在函数体后所在的行。而 Internet Explorer 指出了大括弧前的行。对其他类型的错误消息，如语句 Document.writeln("Hello World");，它导致语法错误，因为 Document 对象输入有误，输入的是大写 D，Navigator 和 Inernet Explorer 都指向相同的行。

除了行号不同，错误消息的解释在 Navigator 和 Internet Explorer 之间也有点不同。注意在图 9-1 中的 Navigator 错误消息中，错误描述是“ missing } after function body ”。Navigator 还显示了在后面应该跟一个大括弧的函数体内最后一条语句。与之相比，图 9-2 中的 Internet Explorer 错误描述是“ Expected'}' ”，表示在函数内的最后一条语句应该跟一个大括弧，Internet Explorer 错误消息列出了出现错误的代码所在行内的字符。Internet Explorer 内的错误消息还让确定是否禁止当前页的脚本。若单击 Internet Explorer 错误消息内的“ Yes ”按钮，那么将禁止当前页的脚本直到单击刷新按钮重载此页。

提示：在下一节中，将学习怎样使用 Microsoft 的脚本调试工具改变 Internet Explorer 中的错误消息来帮助调试 JavaScript 程序。

不管使用的是哪种浏览器，错误消息仅被用来查找程序中错误消息的大致位置而不作为错误的精确指示器。不要一直假设错误消息所指定的行就是程序中的真正错误。例如，在下面代码中的 var result = amount * percentage;语句将导致运行时错误，因为它不能定位 percentage 和 amount 变量。因为 percentage 和 amount 变量不是全局的（它们在 variableDeclaration()函数内声明），所以对 calculatePercentage()函数它们是不可见的，并将

导致运行时错误。尽管是 var result = amount * percentage;语句导致运行时错误，因为它试图访问局部于其他函数的变量，代码中真正的错误是 percentage 和 amount 变量不是在全局部分定义的。

```
<SCRIPT LANGUAGE="JavaScript1.2">
function variableDeclarations(){
        var percentage =.25;
        var amount =1600;
}
function calculatePercentage(){
        var result =amount *percentage;
        document.write("Twenty-five percent of 1,600 is ");
        document.write(result);
}
</SCRIPT>
```

注意您不能看到逻辑错误的错误消息，因为计算机并不十分聪明以至于能够识别逻辑中的错误。例如，若使用 for 语句导致了死循环，解释器无法知道是否真正想继续执行 for 语句代码。在本教程的后面部分，将学习怎样跟踪程序的执行流程来定位逻辑错误。

下一步，将使用错误消息来帮助定位 JavaScript 程序中的错误。

提示：对此节中的练习，能够使用 Navigator 或是 Internet Explorer。

使用错误消息来帮助定位 JavaScript 程序中的错误：

1．在浏览器中，从 Data Disk 的 Tutorial.09 文件夹内打开 FavoriteFoods.html 文件。此文件使用数个函数来打印一列风味食物。在打开文件之后，将立即看到错误消息。若正在使用比 Navigator 4 更高的版本，为了查看错误消息请记住在 location 文本框内输入 javascript: 并按下回车键。

2．关闭浏览器并在文本编辑器或 HTML 编辑器中打开 FavoriteFoods.html 文件。

3．定位和更正错误消息指定的错误，接着在 Data Disk 的 Tutorial.09 文件夹内将它存为 FavoriteFoodsDebug.html。若定位错误存在困难，在 Data Disk 的 Tutorial.09 文件夹下有此文件的正确版本——FavoriteFoodsCorrect.html 文件。

4．在浏览器中打开 FavoriteFoodsDebug.html 文件。

5．在文本编辑器或 HTML 编辑器中继续更正错误直到打开此文件时不再出现任何错误消息。

6．关闭 Web 浏览器和编辑窗口。

9.1.3 使用 alert()方法跟踪错误

在使用错误消息确定不了程序中的错误时，或错误是不产生错误消息的逻辑错误，那

么必须跟踪代码。跟踪检查可执行程序中的每条语句。alert()方法是用来跟踪 JavaScript 代码最有用的方法之一。将 alert()方法放置在程序中的不同地方并用它来显示变量、数组或从函数返回的值的内容。使用此种技术，能够在程序执行过程中监视它们值的改变。例如，检查图 9-3 中的函数，它计算每周的网费并将它四舍五入为最接近的整数。程序语法正确并且不出现任何错误消息。然而，此函数不能返回正确的结果，应该为 485。相反，函数返回的值是 5169107。

```
function calculatePay(){
        var payRate =15;numHours =40;
        var grossPay =payRate *numHours;
        var federalTaxes =grossPay *.06794;
        var stateTaxes =grossPay *.0476;
        var socialSecurity =grossPay *.062;
        var medicare =grossPay *.0145;
        var netPay =grossPay -federalTaxes;
        netPay *=stateTaxes;
        netPay *=socialSecurity;
        netPay *=medicare;
        return Math.round(netPay);
}
```

图 9-3 存在逻辑错误的 calculatePay()

为了跟踪问题，请将 alert()方法放置在程序中您认为可能存在错误的地方。例如，可能首先想检查在 calculatePay()函数中 grossPay 变量是否计算正确。为了检查程序计算 grossPay 是否正确，请在函数中将 alert()方法插在计算 grossPay 变量的后面，如图 9-4 所示。

```
function calculatePay(){
        var payRate =15;numHours =40;
        var grossPay =payRate *numHours;
alert(grossPay);
        var federalTaxes =grossPay *.06794;
        var stateTaxes =grossPay *.0476;
        var socialSecurity =grossPay *.062;
        var medicare =grossPay *.0145;
        var netPay =grossPay -federalTaxes;
        netPay *=stateTaxes;
        netPay *=socialSecurity;
        netPay *=medicare;
        return Math.round(netPay);
}
```

图 9-4　使用 alert()方法来跟踪程序执行的 calculatePay()的函数

提示：以不同层次的缩排插入跟踪程序执行的 alert()方法来清楚地将它们与实际的程序区分开来是十分有用的。

因为 grossPay 变量正确计算的结果为 600，改变 alert()方法开始检查 netPay 变量的值，接着将此语句移至下一个地方。继续使用此技术直到找到错误。calculatePay()不能正确执行是因为将 stateTaxes、socialSecurity、medicare 变量与 netPay 变量相加的行不正确，因为它们使用乘法组合赋值运算符（*=）而不是减法组合赋值运算符（-=）。程序的正确版本（CalculatePayCorrect.html）位于 Data Disk 的 Tutorial.09 文件夹内。

提示：替代使用 alert()方法，能够使用 Window 对象的 status 属性将信息输出到状态条上。

使用 alert()方法的另一种可选的方法是在整个代码内插入多个 alert()方法在代码执行时检测值。例如，可以使用多个 alert()方法来跟踪 calculatePay()函数，如图 9-5 所示。

```
function calculatePay(){
        var payRate =15;numHours =40;
        var grossPay =payRate *numHours;
alert(grossPay);
        var federalTaxes =grossPay *.06794;
        var stateTaxes =grossPay *.0476;
        var socialSecurity =grossPay *.062;
        var medicare =grossPay *.0145;
        var netPay =grossPay -federalTaxes;
alert(netPay);
        netPay *=stateTaxes;
alert(netPay);
        netPay *=socialSecurity;
alert(netPay);
        netPay *=medicare;
alert(netPay);
        return Math.round(netPay);
}
```

图 9-5　使用多个 alert()方法来跟踪程序执行的 calculatePay()函数

使用多个 alert()方法来跟踪值的一个缺点是为了让代码继续执行必须关闭每个对话框。然而，使用多个 alert()方法有时比移动单个 alert()方法更有效。使用多个 alert()方法来跟踪程序值的关键是在整个程序内选择关键点使用它们。考虑一下具有多个函数但不能正确执行的账目程序。在程序内的关键点插入 alert()方法，如在任何函数返回值或变量被赋给新值的地方。利用此方法，能够算出程序的哪部分包含了错误。在发现错误的大致地方之后，例如在某个特殊的函数内，接着能够集中注意力调试那个函数。

下一步，使用警告对话框来帮助定位 JavaScript 程序内的错误：

1．在浏览器内，从 Data Disk 的 Tutorial.09 文件夹内打开 VitalInfo.html 文件。此文件是一个在线程序，它请求各种个人信息。为每个提示对话框填充合适的信息。在将信息输入到所有的提示对话框之后，将看到警告对话框内显示的信息不正确。在文件的多个地方，变量被赋给了不正确的值。

2．现在在文本编辑器或 HTML 编辑器中打开 VitalInfo.html 文件，但是在浏览器窗口内仍打开此文件。

3．在第一条显示 name 变量的变量声明语句 var name = prompt("Please enter your name. ", "")；后，添加 alert()方法：alert(name);，接着保存文件。

4．切换回浏览器窗口，接着单击“ Reload ”或“ Refresh ”。在第一个文本框内输入信息之后，警告对话框将显示 name 变量。

5. 为了查找错误，切换回文本编辑器并开始移动 alert()方法来查看赋给它的每个变量名。一定将传给 alert()方法的参数的变量名改为想要跟踪的变量名。在找到错误之后，在文件内更改它。注意在文件内可能存在多个错误。文件的正确版本（VitalInfoCorrect.html）位于 Data Disk 的 Tutorial.09 文件夹内。更改过的文件显示的最后一个对话框与图 9-6 类似。

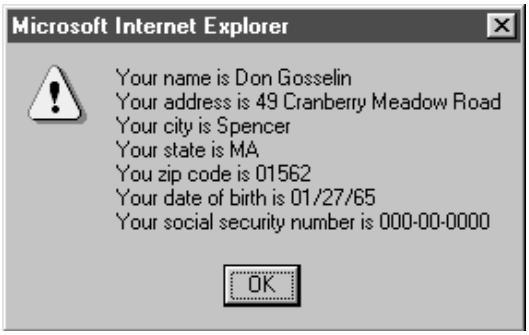


图 9-6　更改过的 VitalInfo.html 文件内的最后一个对话框

6．关闭 Web 浏览器和编辑窗口。

9.1.4　使用 write()和 writeln()方法跟踪错误

在某些情况下想通过一系列的值来跟踪程序中的错误而不是试图一种情况接一种情况地解释在警告对话框中显示的值。能够通过打开新浏览窗口和打印值（使用 write()和 writeln()方法）来创建一系列值。图 9-7 显示了将值打印到窗口中的 calculatePay()函数实例。在函数的开始处使用 valueWindow = window.open("", "", " height=300,width=400");语句来创建新窗口。valueWindow 变量被用来引用新创建的窗口。在整个函数内使用了将值打印到 valueWindow 中的 write()方法。图 9-8 显示了在执行 calculatePay()函数之后 valueWindow 的内容。

```
function calculatePay(){
valueWindow =window.open("","","height=300,width=400");
        var payRate =15;numHours =40;
        var grossPay =payRate *numHours;
valueWindow.document.write("grossPay is "
+grossPay +"<BR>");
        var federalTaxes =grossPay *.06794;
        var stateTaxes =grossPay *.0476;
        var socialSecurity =grossPay *.062;
        var medicare =grossPay *.0145;
        var netPay =grossPay -federalTaxes;
valueWindow.document.write(“netPay minus Federal taxes is "
+netPay +"<BR>");
        netPay *=stateTaxes;
valueWindow.document.write("netPay minus State taxes is "
+netPay +"<BR>");
        netPay *=socialSecurity;
valueWindow.document.write("netPay minus Social Security is "
+netPay +"<BR>");
netPay *=medicare;
valueWindow.document.write("netPay minus Medicare is "
+netPay +"<BR>");
        return Math.round(netPay);
}
```

图 9-7　将值打印到新浏览窗口的 calculatePay()函数

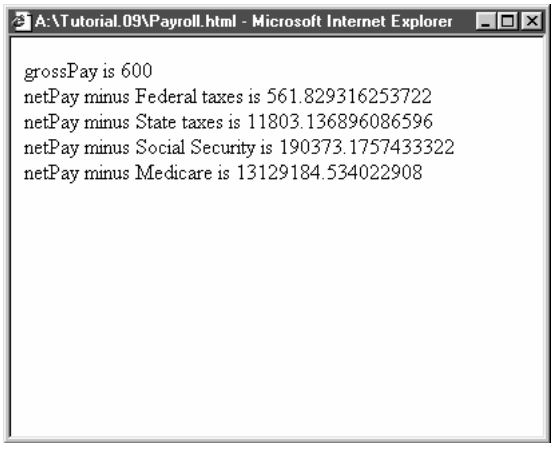


图 9-8　执行 calculatePay()函数之后 valueWindow 的内容

利用在图 9-8 中窗口的内容，能够在值随整个函数的执行而改变时评测每个 calculatePay()函数中的每个变量。

下一步，使用 write()方法和新浏览窗口来帮助确定 JavaScript 程序中的错误：

1．在浏览器中打开 Data Disk 的 Tutorial.09 文件夹中的 Questionnaire.html 文件，并根据每条提示填充相应的信息。文件提示用户输入姓名、地址、城市、州、邮编和电话，并将信息赋给 info 数组，接着将信息打印到屏幕上。然而，首先打印 undefined 并不打印电话号码。

2．在文本编辑器或 HTML 编辑器中打开 Questionnaire.html 文件，但在浏览器窗口中仍打开此文件。

3．在将信息打印到屏幕上的 for 循环之后，添加下面的代码。第一条语句创建一个新的、空的窗口。为了清楚地看到被赋给 info 数组中的每个元素的值，新的 for 循环打印 info 数组的元素编号和每个元素的内容。

```
valueWindow =window.open("","","height=300,width=400");
for(var i =0;i <4;++i){
        valueWindow.document.write("info ["+i +"]="
                +info [i ]+"<BR>");
        }
```

4．保存文件，接着切换到浏览器窗口，并单击“ Reload ”或“ Refresh ”。在文本框中输入信息和将行打印到屏幕之后，将打开一个显示每个数组元素的下标及其内容的窗口。图 9-9 显示了新窗口的输出结果。

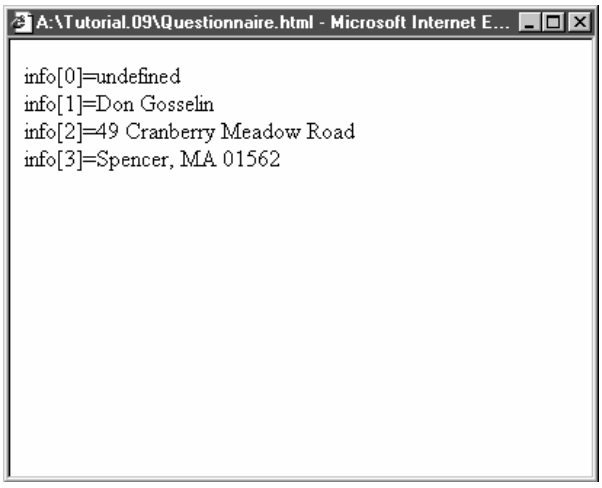


图 9-9　包含数组元素的窗口

5．通过比较输出到新窗口中的结果和被赋给数组元素的值，能够发现和改正文档中错误的源代码。改正过的 QuestionnaireCorrect.html 文件在 Data Disk 上的 Tutorial.09 文件夹内。

6．关闭 Web 浏览器和编辑窗口。

9.1.5 使用注释定位错误

定位 JavaScript 程序中错误的另一种方法是注释可能导致错误的行。可以注释可能导致错误的单行或注释除了已知的正确运行的行之外的所有行。在看到错误消息之后，仅注释掉错误消息行号所指定的语句。保存文档，接着在浏览器中重新打开文档，看是否出现其他错误。若出现其他错误消息，接着也注释掉那些语句。在排除了错误消息之后，检查注释掉的语句查找错误的原因。

提示：在特殊语句中的错误的起因通常是由前面一行代码中的错误引起的。

在图 9-10 中最后 5 条语句被注释掉了，因为它们导致“ yearlyIntrest is not defined ”错误消息。代码的错误是在数条语句内将 yearlyInterest 变量被误拼为 yearlyIntrest。注释掉这些行将有问题的语句隔离开来。

```
document.write("The interest rate for a loan in the amount of "
        +amount +"is "+percentage);
var yearlyInterest =amount *percentage;
//document.writeln("The amount of interest for one year is "
        +yearlyIntrest);
//var monthlyInterest =yearlyIntrest /12;
//document.writeln("The amount of interest for one month is "
        +monthlyInterest);
//var dailyInterest =yearlyIntrest /365;
//document.writeln("The amount of interest for one day is "
        +dailyInterest);
```

图 9-10 使用注释来跟踪错误的代码

尽管图 9-10 中的错误看起来十分简单，但是它将会是遇到的最典型的错误。通常能立即找到错误而不需要注释掉代码或使用其他跟踪技术。然而，在长时间地凝视相同的代码时，简单的拼写错误（如 yearlyIntrest）并不容易被发现。注释掉已知的引起麻烦的行是一个不错的、可以帮助隔离甚至纠正最简单的错误的技术。

可以结合不同的调试技术来帮助查找错误。图 9-11 使用 calculatePay()函数实例来演示怎样将注释和警告对话框结合起来跟踪错误。var grossPay = payRate * numHours;语句是函数中正确执行的最后一条语句。因此在这条语句后的所有语句都被注释掉。接着使用警告对话框来检测每条语句的值，根据程序的流程，按顺序删除每条语句前面的注释，检测和纠正语法错误。

```
unction calculatePay(){
        var payRate =15;numHours =40;
        var grossPay =payRate *numHours;
alert(grossPay);
//      var federalTaxes =grossPay *.06794;
//      var stateTaxes =grossPay *.0476;
//      var socialSecurity =grossPay *.062;
//      var medicare =grossPay *.0145;
//      var netPay =grossPay -federalTaxes;
//      netPay *=stateTaxes;
//      netPay *=socialSecurity;
//      netPay *=medicare;
//      return Math.round(netPay);
}
```

图 9-11　利用注释和 alert()方法跟踪程序执行的 calculatePay()函数

下一步，使用注释来确定 JavaScript 程序中的错误：

1.在浏览器中打开 Data Disk 的 Tutorial.09 文件夹内的 ComparisonExamples.html 文件。文件对两个数字进行了各种逻辑比较。然而，大部分语句包含了错误。在打开文件时将会看到错误消息。此练习的目的是纠正每个错误以便最后的输出结果如图 9-12 所示。

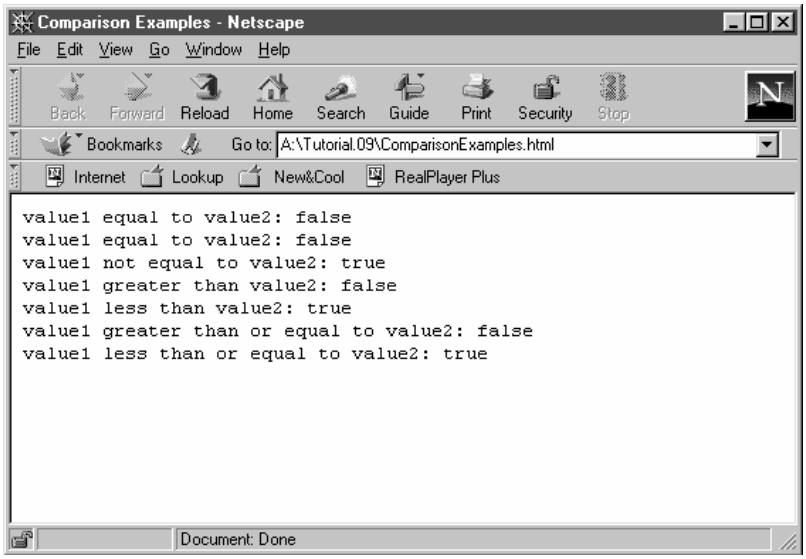


图 9-12　ComparisonExamples.html 的正确输出

2．下一步，在文本编辑器或 HTML 编辑器中打开 ComparisonExamples.html 文件，但是在浏览器中仍打开此文件。

3．开始注释掉脚本段中的所有行。

4．下一步，随机删除某行的注释。每次删除注释时都保存文件，并切换到浏览器，接着重新载入或刷新文件。在遇到错误时纠正每个错误。

5．在纠正完所有错误并且输出如图 9-12 所示之后，关闭浏览器和编辑窗口。文件的正确版本 ComparisonExamplesCorrect.html 文件在 Data Disk 的 Tutorial.09 文件夹内。

9.1.6 其他调试技术

本节的后面部分将讨论确定和纠正 JavaScript 程序中错误的其他方法和技术，包括检查 HTML 代码、分析逻辑、使用 JavaScript URL 测试语句和重新载入 HTML 文档。

检查 HTML 代码

有时不管搜索的时间多长，可能确定不了错误源。在这种情况下，通常是最简单的元素导致了问题。通常它根本不是 JavaScript 代码的问题，而是 HTML 标签的问题。若不能使用本节介绍的任何方法来确定错误，那么接着应该一行一行地分析 HTML 代码，确保所有的标签都有起始尖括号和结束尖括号。另外，确保包括了所有必需的起始和结束标签，如<SCRIPT>和</SCRIPT>标签对。下面是可能导致 JavaScript 错误的 HTML 标签问题的实例，请检查下面的代码。在代码中包含了很难查找到的错误。

```
<HTML>
<HEAD>
<TITLE>Error Example</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2"
      document.writeln(messageVar);
</SCRIPT>
</HEAD>
<BODY>
<H2>Error Example</H2>
</BODY>
</HTML>
```

上面代码中的问题是<SCRIPT>标签漏掉了结束的尖括号。在漏掉结束尖括号的情况下，浏览器看到的仅是在文档的头段中包括的脚本段。因为不执行头段的内容，所以将看不到错误消息，也看不到 writeln()方法的输出。在调试的过程中，可能想是 JavaScript 代码执行不正常而实际上它根本未被执行。

下一步，将查找和纠正 HTML 错误，它导致 JavaScript 程序不能运行：

1．在浏览器内打开 Data Disk 的 Tutorial.09 文件夹内的 PrintData.html 文件。PrintDate.html 文件不能运行。假定它的作用是在警告对话框中显示日期。

2．下一步，在文本编辑器或 HTML 编辑器中打开 PrintDate.html，在浏览器中仍打开此文件。

3．确定和纠正每个 HTML 错误，接着保存文件。切换到浏览器，接着单击“ Reload ”或“ Refresh ”。

4．在纠正错误和将日期打印到屏幕上之后，关闭 Web 浏览器和编辑窗口。文件的正确版本 PrintDateCorrect.html 位于 Data Disk 的 Tutorial.09 文件夹下。

分析逻辑

有时，JavaScript 代码中的错误是逻辑问题，使用跟踪技术很难确定它们。在怀疑代码包含逻辑错误时，必须一种情况接一种情况地分析每条语句。例如，下面的代码包含了逻辑错误，它导致程序不能正常执行。

```
var displayAlert =false,conditionTrue;
if (displayAlert ==true)
        conditionTrue ="condition is true";
        alert(conditionTrue);
```

若执行前面的代码，将永远会看到警告对话框，尽管不应该显示它（displayAlert 被设置为假）。然而，若更仔细地检查 if 语句，将会发现 if 语句在声明 conditionTrue 变量之后就结束了。变量声明后的 alert()方法不是 if 结构的一部分，因为在条件表达式返回真时，if 语句不包括一对花括号来将它所执行的行括起来。alert()方法将显示“ undefined ”值，因为在 if 语句的条件表达式为假时，将“ condition is true ”赋给 conditionTrue 变量的语句被跳过了。为了让代码正常地执行，if 语句必须包括如下花括号：

```
var displayAlert =false,conditionTrue;
if (displayAlert ==true){
        conditionTrue ="condition is true";
        alert(conditionTrue);
}
```

下面的代码是另一个极易被忽略的逻辑错误实例，它使用了 for 语句。

```
var count =0;
for (count =1;count <6;++count);
        document.writeln(count);
```

代码应该将数字 1 到 5 打印到屏幕上。然而，行 for (var count = 1; count < 6; ++count);包含了结束的分号，它标记 for 循环结束了。循环执行了 5 次并将 count 的值变为 6，但是未做其他任何事情，因为在结束的分号前没有任何语句。行 document.writeln(count);是一条单独的语句，它仅执行一次，将数字 6 打印到屏幕上。代码的语法正确，但是不能按照所期望的那样运行。从这些例子可以看出，在代码中忽略细微的逻辑错误是多么地容易。

使用 JavaScript URL 测试语句

若代码中的错误是由单个语句导致的，能够使用 JavaScript URL 来测试此语句而不用重新运行整个程序。在没有 HTML 文档或 JavaScript 源文件的情况下，JavaScript URL 能被用来测试和执行 JavaScript 语句。JavaScript URL 的语法是 javascript:statements(s)。可以在 Navigator 的 Location 框中或在 Internet Explorer 的 Address 框中输入 JavaScript URL，就像是普通的 URL。在浏览器解释 URL 的 javascript:协议时，它执行它后面的 JavaScript 语句。例如，为了在不执行 script 的情况下显示警告对话框，请在浏览器的 Location 或 Address 编辑框中输入 javascript:alert("Hello World")。若使用分号分隔语句，那么在 JavaScript URL 中能够包括多条语句。为了声明变量和在警告对话框中显示它的值，请使用语法 javascript:var stringVar="Hello World";alert(stringVar)。

若试图构造正确的算术表达式，那么 JavaScript URL 十分有用。下面的代码计算 100000 美元抵押物的总利息。计算方法是添加 8%的利息和 35 美元的滞纳金。然而，因为优先级问题代码执行不正确。

```
mortgageBalance = 100000;
mortgageBalance =100000;
interest =.08;
lateFees =35;
document.write(mortgageBalance +lateFees *1 +interest);
```

尽管能够在 JavaScript 程序内直接修改公式结构，但是还能够使用 JavaScript URL 来测试计算方法。下面使用 JavaScript URL 语法在警告对话框内显示公式的结果。为公式添加了括号来改正优先级问题。

```
javascript:mortgageBalance=100000;
interest=.08;lateFees=35;
alert((mortgageBalance +lateFees)*(1 +interest));
```

提示：前面的代码实例被分成了多行是因为文本的空间限制。为了使用 JavaScript URL 来检测代码，在按下回车键之前必须将所有代码输在一行内。

重载 HTML 文档

在编辑 HTML 文档内的 JavaScript 代码时，通常保存文档并在浏览器内单击“ Reload ”或“ Refresh ”来测试修改部分就足够了。然而，使用复杂的脚本，Web 浏览器并不总能完全清空旧错误在内存中的痕迹，尽管已经纠正了错误，理解这一点十分重要。因此，有时需要使用浏览器文件菜单上的打开或打开页命令来完全重新打开文档。然而，有时即使重新打开文件也不能完全清空 JavaScript 旧代码在浏览器中的记忆。若确信已经纠正了代码中的错误，但是程序执行仍不正确，那么请不要忘记执行上面的步骤。

9.1.7　总结

- ✧ 在输入解释器不能识别的代码时将产生语法错误。
- ✧ 若在程序执行时，JavaScript 解释器遇到了问题，那么该问题被称为运行时错误。
- ✧ 运行时错误与语法错误不同，因为它们并不表示 JavaScript 语言错误。相反，在解释器遇到它不能处理的问题时就会产生运行时错误。
- ✧ 逻辑错误是程序设计中的错误，它妨碍了程序按所期望的那样运行。任何程序内的逻辑是按照正确的次序执行不同的语句和过程，它能产生所需要的结果。
- ✧ 在解释器遇到语法或运行时错误时，定位 JavaScript 程序中的错误的措施的第一行是看到的错误消息。
- ✧ 错误消息仅用来确定程序中错误的大致位置，而不能作为错误的精确定位。您不能总假设错误消息所确定的行包含了程序中真正的错误。
- ✧ alert()方法是用来跟踪 JavaScript 代码的最有效的方法之一。可在程序中的不同点插入 alert()方法，并使用它来显示变量或数组的内容，或是函数返回的值。
- ✧ 使用多个 alert()方法来跟踪错误有时比移动单个 alert()方法更有效。
- ✧ 能够创建一系列赋给变量的值，并打开一个新浏览窗口和使用 write()和 writeln()方法输出值。
- ✧ 通常用注释来帮助确定 JavaScript 程序中的错误。能够注释掉可能导致错误的每一行，或是注释掉除已知正确的行之外的所有行。
- ✧ 若使用错误消息跟踪或注释仍不能确定错误，那么请一行一行地分析 HTML 代码，确保所有的标签都具有起始尖括弧和终止尖括弧。
- ✧ 在不编写 HTML 文档或 JavaScript 源文件的情况下，可以使用 JavaScript URL 来测试和执行 JavaScript 语句。
- ✧ 有时为了让修改了的 JavaScript 代码起作用，完全重新打开 HTML 文档或重新启动浏览器是必要的。

9.1.8　问题

1．在输入解释器不能识别的代码时将产生____错误。

a．应用程序

b．逻辑

c．运行时

d．语法

2．若在程序执行时，JavaScript 解释器遇到了问题，那么此问题被称为____错误。

a．应用程序

b．逻辑
c．运行时
d．语法

3．____错误是程序设计中的问题，它妨碍了程序按您所期望的那样运行。
a．应用程序
b．逻辑
c．运行时
d．语法

4．下面哪条语句将导致语法错误？
a．alert("Hello World.");
b．document.writeln("Hello World");
c．Return true;
d．myDate = new Date();

5．假设下面的脚本段是文档中惟一的一个脚本，下面的哪个脚本将会导致运行时错误？

a．
```
<SCRIPT LANGUAGE="JavaScript1.2">
var greeting ="Welcome to my Web page!";
alert(greeting);
</SCRIPT>
```

b．
```
<SCRIPT LANGUAGE="JavaScript1.2">
var greeting ="Welcome to my Web page!";
alert("greeting");
</SCRIPT>
```

c．
```
<SCRIPT LANGUAGE="JavaScript1.2">
var greeting ="Welcome to my Web page!";
alert(""+greeting);
</SCRIPT>
```

d．
```
<SCRIPT LANGUAGE="JavaScript1.2">
alert(greeting);
</SCRIPT>
```

6．下面哪条 if 语句的逻辑不正确？
a．
```
if (count < 5)
document.write(count);
```
b．if (count =< 5)

```
    document.write(count);
c．if (count = 5);
    document.write(count);
d．if (count = 5){
      document.write(count);
    }
```

7．错误消息能够捕获下面的哪种类型的错误？

a．应用程序

b．逻辑

c．运行时

d．断点

8．____是指在执行程序中检测每条语句的行为。

a．追踪（Trailing）

b．跟踪（Tracing）

c．追踪（Tracking）

d．注释

9．在调试中不使用下面的哪种 JavaScript 元素？

a．switch 语句

b．alert()方法

c．write()和 writeln()方法

d．注释

10．下面的哪种代码结构打印文本“ Hello World ”五次？

```
a．for (var count =1;count <6;++count);
    document.writeln("Hello World");
    document.writeln("Hello World");
    document.writeln("Hello World");
    document.writeln("Hello World");
    document.writeln("Hello World");
b．for (var count =0;count <6;++count)
    document.writeln("Hello World");
c．for (var count =0;count <6;++count){
        document.writeln("Hello World");
    }
d．for (var count =0;count <6;++count);
     document.writeln("Hello World");
```

11．下面哪条是执行 JavaScript URL 的正确语法？

a．javascriptURL:alert("Hello World");
b．javascript:"alert('Hello World');"
c．javascript:"alert("Hello World");"
d．javascript:alert("Hello World");

9.1.9 练习

Data Disk 的 Tutorial.09 文件夹内包含了一些在本教程前面所创建的程序。然而，所有的程序都含有错误。请使用在本节中所学的某种调试技能来纠正错误。可以回顾前面的教程确定程序应该怎样运行——但是不要拷贝和复习正确的语法。将这些练习作为一个测试和提高调试技能的机会。在程序文件名的后面都添加了创建的程序所在的章节。在纠正每个文件之后，在 Data Disk 的 Tutorial.09 文件夹内保存它，但是请用_Fixed 代替文件名中的_Tutorial0x 部分。需要纠正的文件如下：

- GreetVisitor_Tutorial02.html
- PoliticalSurvey_Tutorial02.html
- TwoFunctionsProgram_Tutorial02.html
- CarpetCost_Tutorial03.html
- ConvertTemperature_Tutorial03.html
- DailySpecials_Tutorial03.html
- Regions_Tutorial04.html

9.2 高级调试技术和资源

本节目标

在本节将学习：

- 怎样使用 for…in 语句检查对象属性
- 怎样在 Navigator 中使用查看点
- 怎样使用 Netscape JavaScript 调试器
- 怎样使用 Microsoft 脚本调试器
- JavaScript 语言错误和调试资源

9.2.1 使用 for…in 语句检查对象属性

有时程序的错误是由使用了错误的对象属性或将错误的值赋给了对象属性。回顾一下

for…in 循环语句，它为对象内的所有属性执行相同的语句或命令块。能够使用 for…in 循环来判断是否为对象属性赋给了正确的值。在对象具有许多属性，或是在您不能跟踪为什么会给属性赋给了错误的值时，此技术十分有用。考虑下面的构造函数，用来它初始化 Car 对象。

```
function Car(make,model,color){
        this.car_make =make;
        this.car_model =model;
        this.car_color =color;
}
```

在使用语句 myCar = new Car("Jeep", "Gray", "CJ7");初始化新的 Car 对象时，看到车辆的颜色被赋给了 car_model 属性，而车辆的模型被赋给了 car_color 属性。为了跟踪此错误，可以使用 for…in 语句来遍历 Car 对象的所有属性并在警告对话框中显示它们的值。

```
myCar =new Car("Jeep","Gray","CJ7");
var propertiesList ="";
for (prop in myCar){
        propertiesList +=prop +"="+myCar[prop]+"\n";
}
alert(propertiesList);
```

前面的代码将显示如图 9-13 所示的对话框。

图 9-13　使用 for…in 语句创建的警告对话框

从列在警告对话框中的值，可以看到 car_color 和 car_make 被赋给了错误的值，因为在初始化 myCar 对象的语句中的参数顺序不对。代替 myCar = new Car("Jeep", "Gray", "CJ7");，语句应是 myCar = new Car("Jeep", "CJ7", "Gray");。尽管此例使用 for…in 语句来跟踪属性十分简单，但是它让您理解了怎样使用此技术来确定对象属性赋值中的错误。

9.2.2　Navigator 中的查看点

监视对象属性的另一项技术是使用查看点。查看点是一种特定的对象属性，可以在程

序执行过程中监视它的变化。Navigator 包括两种方法：watch()和 unwatch()，它们被用来设置和移去特定对象的观察点。watch()方法为对象属性设置观察点。为了观察特定对象的属性，可以将 watch()方法添加到对象名后并将此语句放在初始化对象的语句后。watch()方法接收两个参数：引号括起来的所查看的属性名和在属性改变时所执行的函数名。watch()方法的语法是对象.watch("属性",函数名);。例如，为了为 myCar 对象的 car_color 属性设置执行 watchColor 函数的观察点，可以在初始化 Car 对象 myCar 的行后使用 myCar.watch("car_color",watchColor);语句。注意 watch()语句中的 watchColor()函数并未像其他大多数函数那样带有括弧。将 watch()方法中所调用的函数的括弧去掉，与将对象方法添加到构造函数时去掉对象方法的括弧相似。

提示：在本节的后续部分，将学习怎样使用 Netscape JavaScript 调试器和 Microsoft 脚本调试器来设置查看点。

在 watch()方法中调用的函数自动传入三个参数：属性名、属性的初始值和赋给属性的新值。在函数的首部需要包括三个参数用来接收参数。下面的函数是在任何时候 car_color 属性改变时所执行的 watchColor()函数。

```
function watchColor(prop,oldValue,newValue){
    alert("The "+prop +"property has changed from "
        +oldValue +"to "+newValue);
    return newValue;
}
```

注意 watchColor()函数的首部包括了三个变量：prop、oldValue、newValue，用它们来接收属性名、旧值和新值参数。另外，请注意函数返回了 newValue 变量。必须返回一个值，否则所观察的属性将被赋给 undefined 值。图 9-14 显示了监视 car_color 属性的完整程序。

注意在图 9-14 中 watch()方法语句放在声明 myCar 对象的语句的后面。将 watch()方法放在对象声明的前面将导致错误。执行程序时将显示三个警告对话框，每次 car_color 属性改变时都显示一个。

unwatch()方法取消先前声明的 watch。unwatch()方法的语法是对象.unwatch("属性");。例如在图 9-14 中，若想在 car_color 属性变为 black 之后取消图 9-14 中的 watch()方法，那么在 car_color 属性变为 red 语句之前添加 myCar.unwatch("car_color");。

在 Navigator 中使用查看点监视变量的变化：

1．在 Navigator 中，从 Data Disk 的 Tutorial.09 目录下打开 FootballTeams.html 文件。FootballTeams.html 在 FavoriteTeam 构造函数中创建对象，接着打印此对象的属性。

2．下一步，在文本编辑器或 HTML 编辑器中打开 FootballTeams.html 文件，但是在浏览器窗口仍打开文件。添加 watch 语句在 favoriteTeam 对象的 teamName 属性改变时进行监视。

3.在初始化 favoriteTeam 对象的语句(favoriteTeam = new FootballTeam("Patriots", "New

England")后，添加 favoriteTeam.watch("teamName", watchName);语句。

```
Function Car(make,model,color){
        this.car_make =make;
        this.car_model =model;
        this.car_color =color;
}
function watchColor(prop,oldValue,newValue){
        alert("The "+prop +"property has changed from "
                +oldValue +"to "+newValue);
        return newValue;
}
myCar =new Car("Jeep","CJ7","Gray");
myCar.watch("car_color",watchColor);
myCar.car_color ="blue";
myCar.car_color ="black";
myCar.car_color ="red";
```

图 9-14　使用 watch()方法监视的 myCar 对象的 car_color 属性

4．下一步，在 FootballTeam()构造函数前添加下面的 watchName()函数。在每次 teamName 属性改变时都将调用 watchName()函数。

```
function watchName(prop,oldValue,newValue){
        alert("The "+prop +
        "property has changed from "
        +oldValue +"to "+newValue);
        return newValue;
}
```

5．保存文件，返回到 Navigator，接着重新载入 FootballTeams.html 文件。应该看到两个对话框，每次 teamName 属性改变时显示一个。

6．关闭 Web 浏览器和编辑窗口。

9.2.3　Netscape JavaScript 调试器

许多高级编程语言（如 Visual C++）都具有调试能力，它们被直接嵌在它们的开发环境中。这些内建的调试能力为跟踪错误提供了更丰富的命令。除了文本编辑器或 HTML 编辑器，JavaScript 编程语言未提供开发环境。若不计 watch()和 unwatch()（Navigator 特有的）方法，那么在 JavaScript 中惟一的真正的调试工具是浏览器创建的错误消息。为了对 JavaScript 提供调试能力，Netscape 和 Microsoft 开发的调试工具都能与浏览器一起使用来

调试 JavaScript 代码。首先，将讨论 Netscape 工具——JavaScript 调试器。

目前，已经学习了怎样解释错误消息和纠正导致错误的语句。尽管它们十分有用，但是仅在解析语法和运行错误时错误消息才有用。还学习了帮助确定逻辑错误的某些技术。在遇到逻辑错误时手工地检查代码通常是所采取的第一步，或者是使用警告对话框来跟踪值。这些技术对小的程序很有效。然而，在创建含有多个对象、方法和函数的大型程序时，确定逻辑错误将十分困难。例如，在程序中有一个函数，它调用数个不同的构造函数来初始化对象。接着每个被初始化的对象可能调用它父辈对象（或其他祖先对象）的方法或使用它的属性。试图使用简单的工具如警告对话框来跟踪这样程序的逻辑和流程将十分困难。Netscape 提供了多种工具来帮助跟踪每行代码，它提供了一种查找和解析逻辑错误的更有效的方法。

提示：在 Navigator 4.5 和高版本中，在 Navigator 的 Location 文本框中输入 javascript:，且没有任何代码跟在它的后面，那么在状态条中显示 JavaScript 错误时，将会打开显示浏览器所发现的错误的错误页。

在能够使用 JavaScript 调试器之前，必须从 http://developer.netscape.com/software/tools/index.html?content=/software/jsdebug.html 下载并安装它。按照下面的命令安装程序。在安装的过程中，将会提示是否同意许可。若同意，单击“ Certificate ”按钮确保 Netscape 签署了此文档。若未被授予访问的特权，那么将不能使用 JavaScript 调试器。注意若未单击“ Remember ”确定按钮，那么每次打开调试器时都将提示是否同意许可。

提示：在安装 JavaScript 调试器的过程中不要修改 Navigator 中的页面，否则安装将失败！

JavaScript 调试器是一个小应用程序，在下载过程中将自动启动它。为了在后续的 Navigator 会话中启动调试器，必须打开\JSDebug 文件夹下的 Jsdebugger.html 文件，此文件夹安装在计算机的 Netscape 文件夹下。因此，在安装完之后，应该立即为 Jsdebugger.html 文件创建书签。打开 Jsdebugger.html 文件将会创建三个窗口：含有 JavaScript 调试器启动页的 Navigator 窗口、执行 JavaScript 调试器小应用程序的 Navigator applet stub 窗口以及含有 JavaScript 调试器程序的小应用程序窗口。打开需要在含有 JavaScript 调试器启动页的窗口中调试的文档。在小应用程序载入之后，可以使用 applet stub 窗口中的 Exit Debugger 链接来关闭小应用程序。

帮助：打开需要标记书签的页面，接着按下“ Ctrl+D ”，就在 Navigator 中创建了书签。

首次在 Navigator 会话内打开 Jsdebugger.html 文件时，通常需要花费几分钟等待小应用程序启动，其间 applet stub 窗口将会显示“ Loading Debugger ”消息。图 9-15 显示了在小应用程序打开时启动页和 applet stub 窗口的样式。图 9-16 显示了在打开需调试的文档之前 JavaScript 调试器小应用程序窗口的样式。

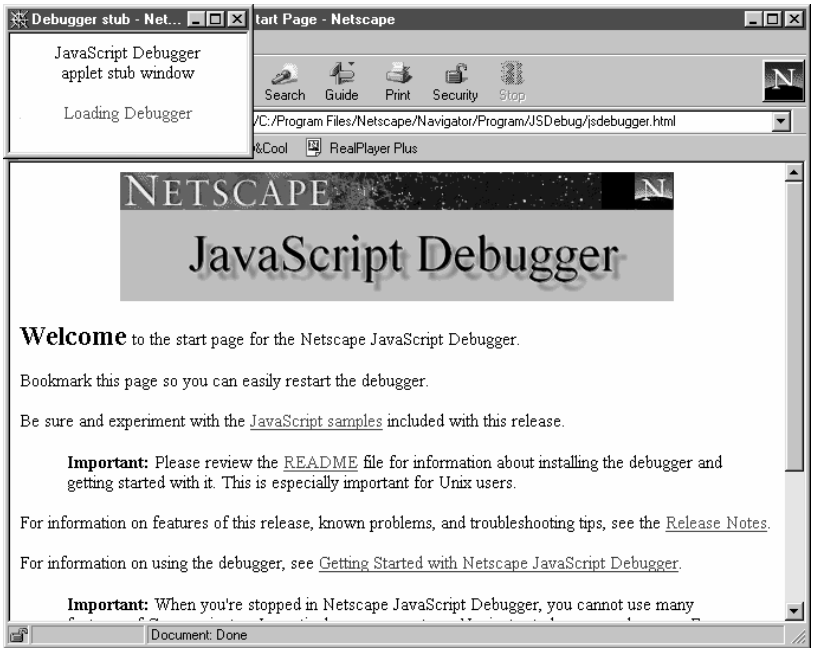


图 9-15　在小应用程序打开时的启动页面和 applet stub 窗口

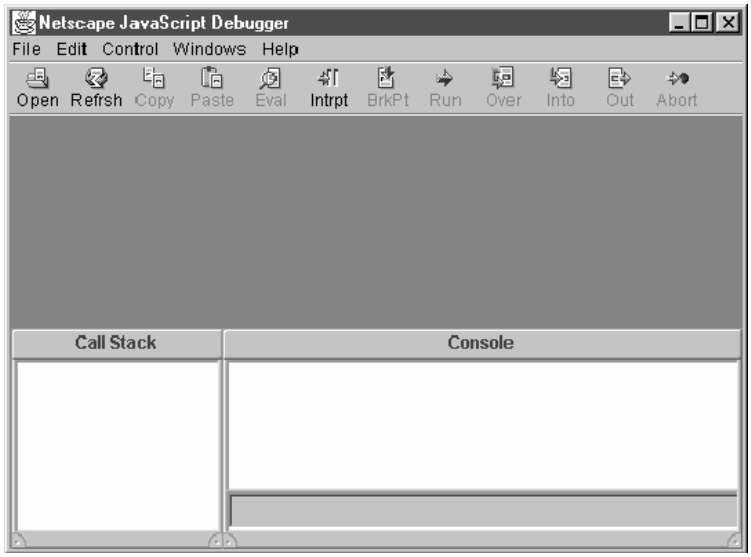


图 9-16　在打开需调试的文档之前 JavaScript 调试器小应用程序窗口

在首次载入小应用程序时，JavaScript 调试器窗口将包含一个调用堆栈窗口和控制台窗口。在本节的后续部分将学习怎样使用这些窗口。

源代码窗口

JavaScript 调试器中的 Control 菜单含有数个用于在中断模式下进行输入和运行的命令。

中断模式可以临时挂起、暂停和执行程序，以便监视值和跟踪程序执行。在进入中断模式和使用 Control 菜单下的任何工具之前，必须在源代码窗口中打开文档。源代码窗口是一个单独的调试窗口，它用于每个在 JavaScript 调试器中打开的 HTML 页面。能够同时打开多个源代码窗口。为了在源代码窗口中打开 HTML 文档，必须：

1．选择 Control 菜单下的 Set Interrupt 命令。Set Interrupt 命令指示 JavaScript 调试器执行到 JavaScript 代码之后立即进入中断模式。

2．切换到含有 JavaScript 调试器启动页的 Navigator 窗口，接着打开想要调试的 HTML 页。在执行到 JavaScript 代码之后，Navigator 立即暂停执行并切换到 JavaScript 调试器。

提示：为了清除为 HTML 文档设置的断点，请选择 Control 菜单下的 Clear Interrupt。

提示：本教程指导使用 JavaScript 调试器菜单来执行命令。然而，许多命令还能够通过 JavaScript 调试器工具条上的图标来调用。

在 JavaScript 调试器中打开文档之后，在左边空白处所显示的颜色和图标指明了 JavaScript 程序不同的元素。例如，在函数体左侧显示的是桔黄色的竖条，黄色的竖条则指出了文档的主<SCRIPT>段。左边空白处的箭头指出了当前调试器暂停执行处的 JavaScript 语句。图 9-17 显示了一个在源代码窗口中显示的 HTML 代码实例。注意文件名被显示在源代码窗口的顶部。

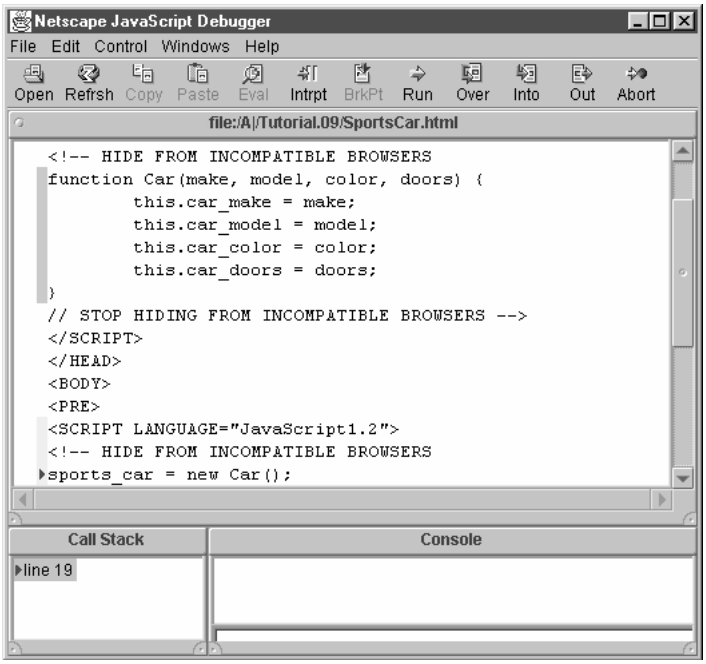


图 9-17　在源代码窗口中打开的 HTML 文档

提示：若想让 JavaScript 调试器在左侧空白处显示行号，那么请先选中 Edit 菜单下的 Preferences，接着选中 Show Line Numbers。可以用选中 Edit 菜单下的 Preferences，然后选

中 Hide Line Numbers 来隐藏行号。

下一步，将在源代码窗口中打开需调试的文档。在本节后续部分所使用的 HTML 文档是 CompanyObjects.html。CompanyObjects.html 含有数个函数和构造函数，它们被用来打印企业的相关信息。此文档未包含任何错误——本节大部分练习的目的是熟练使用 JavaScript 调试器和 Microsoft 的脚本调试器。

帮助：请记住本节内是在 Navigator 4.0 中使用 JavaScript 调试器的步骤。若使用的不是 Navigator 4.0，那么在调试器进入中断模式时，在下面步骤中所引用的语句可能并不直接对应于您所使用的 Navigator 进入中断模式时的位置。

在源代码窗口中打开用于调试的 HTML 文档：

1．启动 Navigator，接着打开 Jsdebugger.html 文件。若需要，请同意许可。

2．在 JavaScript 调试器小应用程序启动之后，选择 Control 菜单下的 Set Interrupt。

3．切换到含有 JavaScript 调试器启动页的 Navigator 窗口并打开 Data Disk 的 Tutorial.09 文件夹内的 CompanyObjects.html 文件。在调试器执行到 JavaScript 代码之后，程序将立即暂停，并且将在源代码窗口中以中断模式打开 CompanyObjects.html 文件。调试器暂停在 Sales.prototype = new Company;行处，因为它是将要执行的第一条 JavaScript 代码。

单步命令

在进入中断模式之后，可以使用 Control 菜单下的 Step Into、Step Over、Step Out 命令。Step Into 命令执行完一行代码后接着就暂停下来直到让调试器继续执行。此特性让您在程序执行时有机会推断程序的流程和结构。

随着使用 Step Into 命令执行整个代码，调试器将在 JavaScript 程序的每个函数内每行上都停下来。在单步执行整个程序来跟踪逻辑错误时，跳过所知的能够正确运行的函数十分方便。Step Over 命令让您跳过函数调用。程序仍然执行所跳过的函数，但是在调试器中它看起来好像是执行一条语句。

Step Out 命令执行当前函数内的所有剩余代码。若函数是从另一个函数内调用的，那么将执行当前函数内的所有剩余代码并且调试器停在调用函数的下一条将执行语句处。

在程序进入中断模式时，并不停止执行程序——它仅仅是被挂起。在程序进入中断模式后要恢复程序的执行，请选择 Control 菜单内的 Run 命令。Run 命令结束调试过程并正常地执行程序的其余部分。还能够选择 Control 菜单中的 Abort 命令来结束调试过程。Abort 命令停止执行当前函数并执行程序的其余部分。

帮助：再次提醒一下，本节中的步骤所使用的是 Navigator 4.0。若在使用单步命令时使用的不是 Navigator 4.0，那么使用的 Navigator 可能不会在下面步骤中所列出的语句处暂停下来。

下一步，练习使用单步命令来跟踪程序的执行：

1．返回到 JavaScript 调试器中的 CompanyObjects.html 文档。

2．选择 Control 菜单下的 Step Into。因为语句 Sales.prototype = new Company;使用

prototype 属性来扩展 Company 对象的定义，所以程序执行流程转移到 Company()构造函数内的第一条语句处。

3．再次选择 Step Into，程序执行 Company()构造函数的第一条语句并将流程转移到第二条语句处。

4．下一步，选择 Control 菜单下的 Step Out。Step Out 命令执行 Company()构造函数内的剩余语句，接着返回到 Sales.prototype = new Company;语句处。

5．选择 Step Into，程序执行至下一条语句 Production.prototype = new Company;，它使用 prototype 属性来扩展 Company 对象的定义。

6．下一步，选择 Control 菜单下的 Step Over。Step Over 命令执行 Company 构造函数内的所有命令，接着移至下一条语句 var personnel;。

7．最后，既然知道程序不存在问题，选择 Control 菜单下的 Run 命令继续运行 JavaScript 代码的剩余部分。

8．关闭 Navigator 和 JavaScript 调试器窗口。

断点

在 JavaScript 调试器中跟踪程序的执行的另一种方法是在代码中插入断点。断点是代码中的一条语句，在此语句处程序流程进入中断模式。在程序暂停在断点后，能够使用 Step Into、Step Over、Step Out 命令来跟踪程序执行，或使用 Run、About 命令来执行整个程序并运行到下一个断点处。多断点为在关键点处（可能存在错误）暂停程序执行提供了一种方便的方法。

打开利用断点调试的 HTML 文档的步骤和利用 Set Interrupt 命令所采取的步骤有点不同。必须在 Navigator 内打开文档，接着在 JavaScript 调试器内再次打开它。详细的步骤如下：

1．启动 Navigator，接着打开 Jsdebugger.html 页面启动 JavaScript 调试器。接着打开需调试的 HTML 文档。

2．切换到 JavaScript 调试器并选择 File 菜单中的 Open 命令。所弹出的打开对话框列出了启动 JavaScript 调试器后已打开过的 HTML 文档。选择在步骤 1 Navigator 中所打开的文档。

3．在打开的源代码窗口内，将光标移至想暂停程序执行的行上，接着选择 Control 菜单下的 Set Interrupt。

4．切换到 Navigator 并单击“ Reload ”按钮或单击执行 JavaScript 代码的表单按钮。

下一步，将练习使用断点：

1． 在 Navigator 内，打开 Jsdebugger.html 页面启动 JavaScript 调试器，接着打开 Data Disk 的 Tutorial.09 文件夹内的 CompanyObjects.html。

2．切换至 JavaScript 调试器并选择 File 菜单下的 Open 命令。所弹出的打开对话框列出了启动 JavaScript 调试器后已打开过的 HTML 文档。从 Open 对话框内选择 CompanyObjects.html，接着单击“ Open ”按钮。

3．确定插入点，它可以是 this.territory ="North America";行内的任何位置，并选择

Control 菜单下的 Set Interrupt。在含有断点的行旁的源代码窗口的空白处将显示一个红圆点。

提示：还能够单击程序暂停执行处行旁的左空白处来插入断点。

4．在 this.facilities = "New York, Chicago, and Los Angeles";行内插入另一个断点。图 9-18 显示了具有两个断点集的源代码窗口的样式。

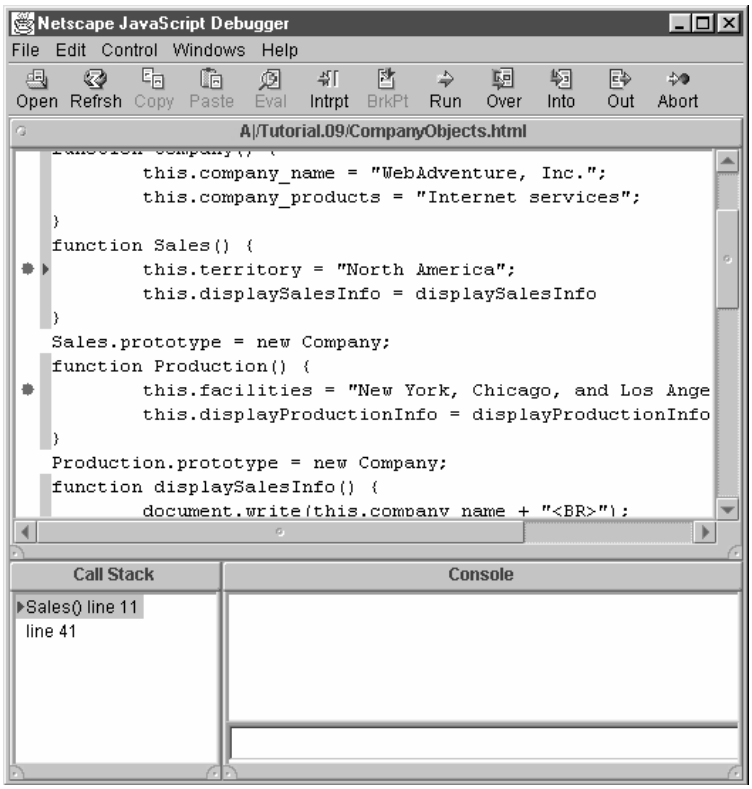


图 9-18　源代码窗口内的断点

5．切换到 Navigator 并单击“Reload”按钮。程序开始运行，接着切换到 JavaScript 调试器并暂停在第一个断点处。

6．选择 Control 菜单下的 Run。执行两个断点之间的语句，接着程序暂停在第二个断点处。

7．在此处，能够再次选择 Run 命令来继续执行程序，或设置其他断点，或是使用任何其他的调试命令。

删除断点：

1．将光标移至含有第一个断点的行内的任何位置并选择 Control 菜单下的 Clear Breakpoint。源代码窗口空白处的红圆点将被删除掉。

提示：还能够单击源代码窗口空白处内的红圆点来删除断点。

2．重复步骤 1，删除文件内的第二个断点。

提示：能够在断点对话框内添加和删除断点。为了弹出断点对话框，选择 Window 菜单下的 Breakpoints。还能够使用 Breakpoints 对话框来设置程序在断点处暂停的条件。例如，能够设置仅当变量的内容与给定的值匹配时才设置断点。

跟踪变量和表达式

在使用单步命令和断点来跟踪程序执行时，可能还需要在程序执行过程中跟踪变量和表达式是怎样变化的。例如，可能存在 resultNum = firstNum / secondNum;语句。您知道此语句会导致除零错误，但是不知道何时 secondNum 变为 0。跟踪程序执行和确定 secondNum 变为 0 的位置的能力能够查明逻辑错误的原因。在调试过程中使用控制台窗口、查看窗口和 Evaluate 命令能够监视中断模式下的变量和表达式。

提示：使用监视窗口能够监视 JavaScript 调试器内的对象的属性。要显示监视窗口，请选中 Windows 菜单下的 Inspector 菜单项。通过在源代码窗口中突出对象和属性以及选中 Edit 菜单下的 Inspect 还能够显示特殊对象的属性值。

在调试 HTML 文档时，控制台窗口（自动显示在 JavaScript 调试器的右下角）的上部显示了变量和表达式的值。当 JavaScript 调试器处于中断模式时，可以通过突出变量或表达式和选中 Edit 菜单下的 Evaluate 来显示它们的值。控制台的下部允许在调试过程中修改变量的值。图 9-19 显示了在突出和对表达式求值后的 JavaScript 调试器的样式。

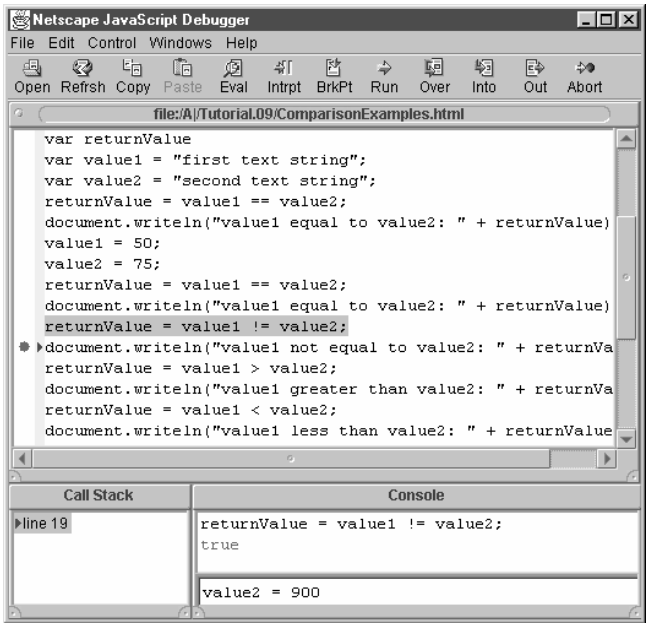


图 9-19　控制台窗口

提示：在调试过程中，选中 Edit 菜单下的 Clear Console 可以清除控制台窗口中的值和

结果。

查看窗口用于监视输入的特定变量和表达式。例如，能够在查看窗口中输入变量名或表达式来监视变量或表达式在程序执行过程中是怎样变化的。还能够在查看窗口中输入自定义的表达式来观察在程序执行时它们值的变化。例如，若在查看窗口中输入 sampleVariable * 2 表达式，那么在 sampleVariable 变化时表达式的值也随之改变。

将变量或表达式添加到查看窗口中的最简便的方法是在中断模式下突出变量或表达式，接着选中 Edit 菜单下的 Copy to Watch。还可以通过选中 Windows 菜单下的 Watches 来显示查看窗口。图 9-20 显示了查看窗口的一个实例。

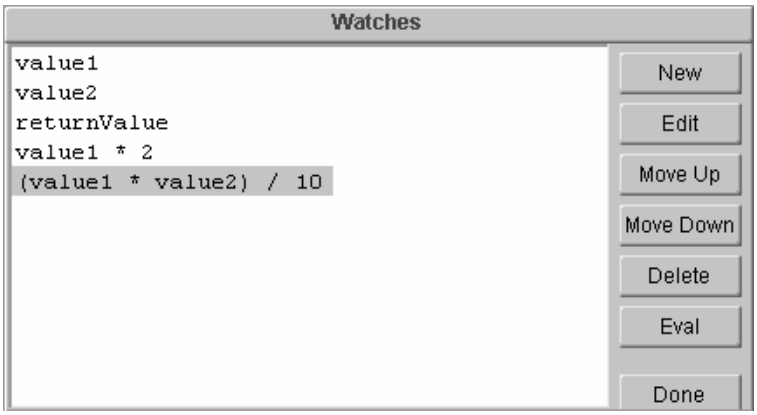


图 9-20　查看窗口

下一步，练习跟踪变量：

1．切换到 JavaScript 调试器中的 CompanyObjects.html 文档。

2．在文档体的脚本段中，突出 var personnel;语句并选中 Edit 菜单下的 Copy to Watch。注意此时 personnel 变量的值为 null，因为在声明中没有给它赋初值。

3．在 displaySalesInfo() 函数内的 personnel = 50; 语句处插入断点，并在 displayProductionInfo()函数内的 personnel = 100;语句处插入另一个断点。

4．切换到 Navigator 并单击“ Reload ”按钮。程序开始执行，接着在第一个断点处停止下来。

5．选中 Control 菜单下的 Step Into 来执行 personnel = 50;语句。personnel 变量的新值将被打印至控制台窗口。

6．下一步，选中 Control 菜单下的 Run。程序接着执行并在下一个断点处停止下来。选中 Control 菜单下的 Step Into 执行 personnel = 100;语句。personnel 变量的新值将被打印至控制台窗口。

7．现在，将光标移至控制台窗口的下部并输入 personnel = 200，接着按下回车键。personnel 变量的值变为 200 并被打印至控制窗口的上部。

8．选中 Run 命令执行下面的程序。

调用堆栈窗口

在运行含有多个函数的 JavaScript 程序时，计算机必须记住执行它们的顺序。例如，若在 accountsPayable()函数中调用了 accountsReceivable()函数，那么计算机必须记住在执行完 accountsReceivable()函数后立即返回至 accountsPayable()函数。同样，若在 accountsReceivable()函数被 accountsPayable()函数调用之后，accountsReceivable()函数又调用了 depositFunds()函数，那么计算机必须记住在执行完 depositFunds()函数时返回至 accountsReceivable()函数，接着在执行完 accountsReceivable()函数之后立即返回至 accountsPayable()函数。调用堆栈是指过程（如函数、方法或事件处理器）在程序中执行的顺序。每次程序执行一个过程，过程都将被添加到调用堆栈的顶部，接着在它执行完之后就立即删除它。

在跟踪含有多个函数的大程序中的逻辑错误时查看调用堆栈内容的功能十分有用。例如，您可能在多个函数之间将变量作为参数传递。在某个地方，变量被赋给了错误的值。查看调用堆栈（同时使用跟踪和 Evaluate 命令）使得定位导致问题的函数变得十分容易。在首次启动 JavaScript 调试器时将显示调用堆栈窗口。当进入中断模式调试程序时，在单步运行程序的过程中，过程自动地被添加到调用堆栈和自动地从调用堆栈中删除。

下一步，将单步执行 CompanyObjects.html 内的某些函数来观察调用堆栈窗口的内容。

帮助：若使用的不是 Navigator 4.0，那么在使用调用堆栈窗口和单步命令时，Navigator 在调用堆栈窗口中显示的行号可能与下面步骤中列出的行号不一样。

观察调用堆栈窗口的内容：

1．切换回 JavaScript 调试器中的 CompanyObjects.html 文档。

2．清除在上一次练习中设置的断点，接着选中 Control 菜单下的 Set Interrupt。

3．切换回 Navigator 并重新载入文档。程序开始执行，接着切换至 JavaScript 调试器并在第一条语句处停止下来。注意行 14 被添加到调用堆栈窗口。它是当前调用堆栈内的第一个调用。

4．选中 Control 菜单下的 Step Into。程序控制转移到 Company()函数，Company()行 7 被添加到调用堆栈窗口，位于第一条调用的上面。现在它是当前调用堆栈内的第一个调用。

5．选中 Control 菜单下的 Step Out。执行 Company()函数的剩余部分，程序控制返回到行 14，并且从调用堆栈中删除了 Company()行 7。

6．再次执行 Step Into 命令，调用堆栈中的行 14 被行 19（它是下一条执行的语句）替换了。再次选择 Step Into 命令又将 Company()函数添加至调用堆栈，因为此函数被行 19 语句调用。

7．选择 Run 命令执行程序的下面部分。

8．关闭 Navigator 和脚本调试器。

9.2.4 Microsoft 脚本调试器

Microsoft 脚本调试器与 Netscape 的 JavaScript 调试器令人惊讶地相似。两个应用程序

的一个主要不同之处在于脚本调试器不仅能够调试 JavaScript 程序，而且还能够调试 VBScript、Java 小应用程序、Java Bean 和 ActiveX 组件。相比之下，JavaScript 调试器仅能够调试 JavaScript 和 Java 程序。本节将只讨论怎样调试 JavaScript 程序。注意脚本调试器将 JavaScript 称为 JScript。本教程始终使用语言的本名——JavaScript（如 Netscape 提出的）。

可以从 Microsoft Windows 脚本技术网址 http://msdn.microsoft.com/scripting/default.htm 下载脚本调试器。在安装此程序之前，一定要完整地阅读 Microsoft 的安装步骤，特别是那些使用 Windows 95 或 Windows 98 的用户。用于上述操作系统的脚本调试器存在错误，需要特殊的安装过程才能使程序正确地运行。在 Microsoft Windows 脚本技术网址下也能够获得安装脚本调试器的步骤。需要指出的是，在撰写本教程时，脚本调试器的最新版本——1.0 版不是一个稳定的程序并且包含错误，可能导致系统错误。若使用的是脚本调试器并且它不能正常运行，那么可能需要重新启动 Internet Explorer 或完全重启系统。

由于程序的不稳定，可能不能实现本节内介绍的所有步骤。不管程序的不稳定性，下面的步骤描述了使用脚本调试器调试 JavaScript 代码的通用的和有用的方法。

脚本调试器窗口

不需要像使用 JavaScript 调试器那样打开一个单独的 HTML 文档来启动脚本调试器。相反，在 Internet Explorer 中打开想要调试的文档并使用查看菜单下的脚本调试程序子菜单。脚本调试器的子菜单包括两个命令：打开和在下一条语句中断。打开命令在当前脚本调试窗口内打开当前文档。在下一条语句中断与 JavaScript 调试器的 Set Interrupt 命令相同。它指示脚本调试器在遇到 JavaScript 代码时立即进入中断模式。在脚本调试器窗口内打开 HTML 文档和在第一条语句处进入中断模式的步骤如下：

1．在 Internet Explorer 中打开 HTML 文档，接着选中查看菜单下的脚本调试器程序子菜单中的再下一条语句中断。此命令指示 Internet Explorer 您想调试的当前文档。

2．在 Internet Explorer 中，单击刷新按钮或单击执行 JavaScript 代码的表单按钮。在执行第一条 JavaScript 语句时将在脚本调试器窗口中打开文档。

提示：此教程教导使用脚本调试器菜单来执行命令。然而许多命令能够使用脚本调试器工具条上的图标来调用。

另外，可以在脚本调试器窗口内打开 HTML 文档并选中脚本调试器调试菜单下的再下一条语句中断。接着，切换回 Internet Explorer 并单击刷新或执行 JavaScript 代码的表单按钮。

一旦在脚本调试器中打开了文档，将立即在左空白处显示不同图标来表示不同的元素。例如，黄色的箭头用来表示下一条执行的语句。代码元素的不同类型是通过语法颜色来区分的。颜色编码使理解 JavaScript 程序的结构和代码十分容易。例如，用于 JavaScript 关键字默认的语法颜色是蓝色。图 9-21 显示了脚本调试器窗口中的处于中断模式下的文档实例。

提示：您的脚本调试器窗口可能与图 9-21 实例不同，因为它不是自动打开调用堆栈和命令窗口。可以通过选中查看菜单下的调用堆栈或命令窗口来显示调用堆栈或命令窗口。

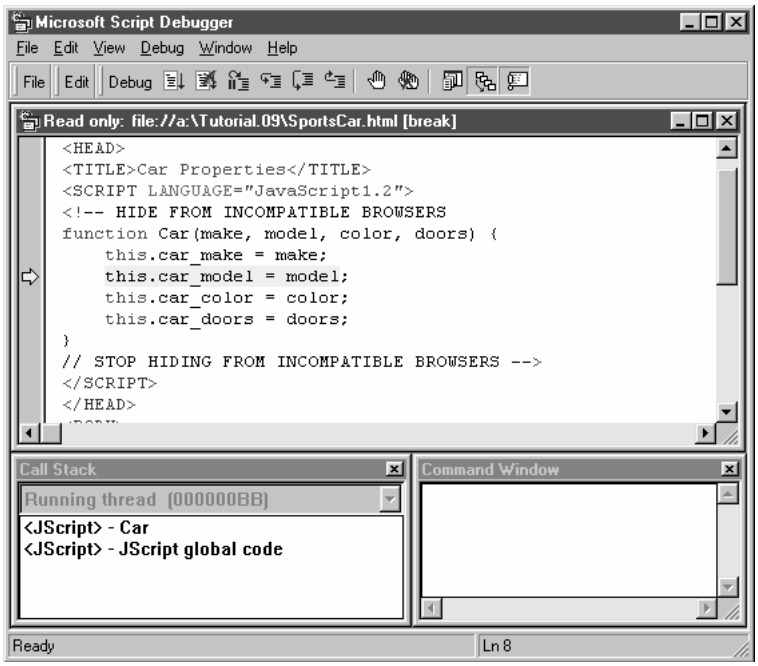


图 9-21 脚本调试器窗口中的处于中断模式下的 HTML 文档

在源代码窗口中打开用于调试的 HTML 文档：

1．在 Internet Explorer 中，打开 Data Disk 的 Tutorial.09 文件夹内的 CompanyObjects.html 文件。

2．选中查看菜单下脚本调试程序子菜单中的再下一条语句中断。

3．单击刷新按钮。在执行第一条 JavaScript 语句 Sales.prototype = new Company;时将在脚本调试器窗口中打开文档。

单步命令

脚本调试器中的 Step Into、Step Over、Step Out 命令与 JavaScript 调试器中的对应的命令相同。它们用于在进入中断模式后继续执行程序。在脚本调试器中，这些命令位于调试菜单中。要在进入中断模式后恢复程序的执行，请选中调试菜单中的运行。调试菜单中的 Run 命令结束调试过程并正常地执行程序的下面部分。能够选中调试菜单中的中止调试命令来结束脚本调试器中的调试过程。

下一步，练习使用单步命令来跟踪程序的执行：

1. 返回至脚本调试器窗口内的 CompanyObjects.html 文档。

2．选中调试菜单中的 Step Into。因为 Sales.prototype = new Company;语句使用 prototype 属性来扩展 Company 对象的定义，所以程序流程转移至 Company()构造函数内的第一条语句。

3．再次选中 Step Into，程序执行 Company()构造函数内的第一条语句并移至第二条语句。

4．下一步，选中调试菜单中的 Step Out。Step Out 命令执行 Company()构造函数内的

剩余语句，接着移至下一条语句 Production.prototype = new Company;，它使用 prototype 属性来扩展 Company 对象的定义。

5．下一步，选中调试菜单中的 Step Over。Step Over 命令执行 Company 构造函数内的所有命令，接着移至下一跳语句 sales_object = new Sales();。

6．最后，既然知道程序不存在问题，那么选中调试菜单中的 Run 来完成执行 JavaScript 代码的剩余部分。

7．关闭脚本调试器，但是在 Internet Explorer 中仍打开 CompanyObjects.html。

断点

在脚本调试器中使用断点与在 JavaScript 调试器中使用断点非常相似。断点命令位于调试菜单中，打开调试的文档有点不同。在脚本调试器中打开想使用断点调试的 HMTL 文档的步骤如下：

1．在 Internet Explorer 打开想调试的 HTML 文档，选中 View 菜单下脚本调试程序子菜单中的再下一条语句中断，接着单击“ Refresh ”按钮。

2．在脚本调试器中，将光标移至想暂停程序执行的行上并选中 Debug 菜单下的切换断点。

3．选中 Debug 菜单下的 Run 来执行代码直至断点。

下一步，练习使用断点：

1．返回至 Internet Explorer 中的 CompanyObjects.html 文档。

2．选中 View 菜单下脚本调试程序子菜单中的再下一条语句中断，接着单击“ Refresh ”按钮。

3．当脚本调试器在第一条语句处暂停之后，在行 this.territory = "North America";中的任何地方定位插入点，并选中 Debug 菜单下的切换断点。在紧邻含有断点的行、脚本调试器窗口左侧的空白处将显示一个红圆点。

4．在行 this.facilities = "New York, Chicago, and Los Angeles";中添加另一个断点。图 9-22 显示了具有两个断点的脚本调试器窗口的样式。

5．选中 Debug 菜单下的 Run。程序开始运行，接着在第一个断点处暂停下来。

6．再次选中 Debug 菜单下的 Run。执行两个断点之间的语句，接着程序在第二个断点处暂停下来。

7．现在，可以再选中 Run 命令来继续执行程序，或设置其他断点，或使用任何其他调试命令。

删除断点：

1．将光标移至包含了第一个断点的行内的任何地方，并选中 Debug 菜单下的 Toggle Breakpoint。在脚本调试器窗口左侧空白处的红圆点将消逝。

2．重复步骤 1 删除文件中的第二个断点。

3．选中 File 菜单中的 Exit 关闭脚本调试器。

提示：可以选中 Debug 菜单中的 Clear All BreakPoints 来删除脚本调试器中 HTML 文档内的所有断点。

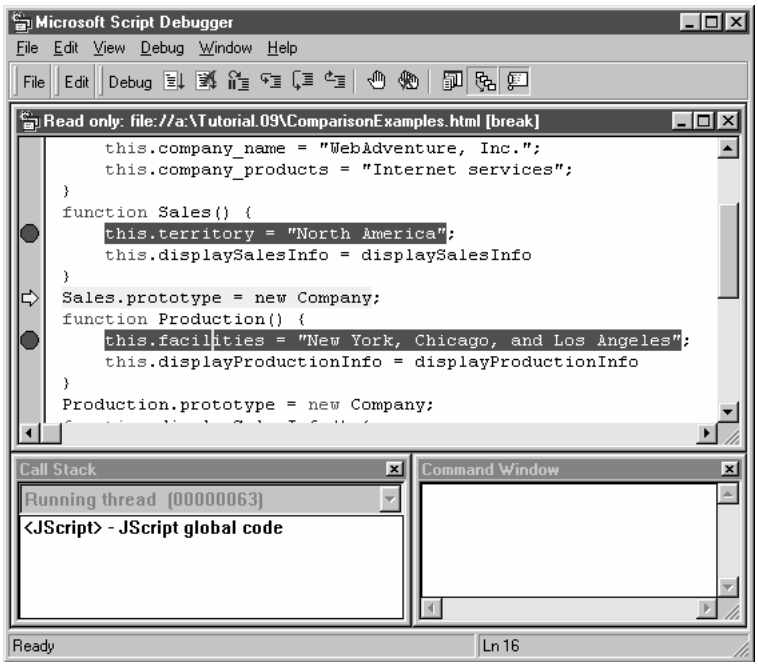


图 9-22 在脚本调试器窗口中的断点

跟踪变量和表达式

脚本调试器和 JavaScript 调试器之间的主要区别在于脚本调试器并不含有查看窗口，它用来在程序执行的过程中监视变量或表达式是怎样改变的。脚本调试器只能使用命令窗口，它与 JavaScript 调试器的控制台窗口相似。

命令窗口不像 JavaScript 调试器控制台窗口那样分成上面部分和下面部分。相反，命令窗口由一个单独的区域组成，在调试过程中处于中断模式时能够在其中查看和修改变量或表达式的值。要在命令窗口中显示变量或表达式的值，请输入变量或表达式并按下回车键。它们的值直接在命令窗口中变量或表达式的下面打印出来。要改变变量的值，请在命令窗口中输入变量名，接着输入等号和新值，并按下回车键。新值将在所输入的语句下面打印出来。图 9-23 中的命令窗口包含了怎样查看变量的实例以及怎样改变值的实例。

提示：命令窗口还允许在调试过程中改变对象的属性值。例如，若想在调试过程中将 myCar 对象的 car_color 属性修改为 blue，请在命令窗口中输入 myCar.car_color = "blue"并按下回车键。

下一步，练习跟踪变量：

1．切换至 Internet Explorer 中的 CompanyObjects.html 文档。

2．选中 View 菜单下的脚本调试程序子菜单中的 Break at Next Statement，接着单击“ Refresh ”按钮。

3．在脚本调试器在第一条语句暂停下来之后，在 displaySalesInfo()函数内的 personnel = 50;语句中插入一个断点，并在 displayProductionInfo()函数内的 personnel = 100;语句中插

入另一个断点。

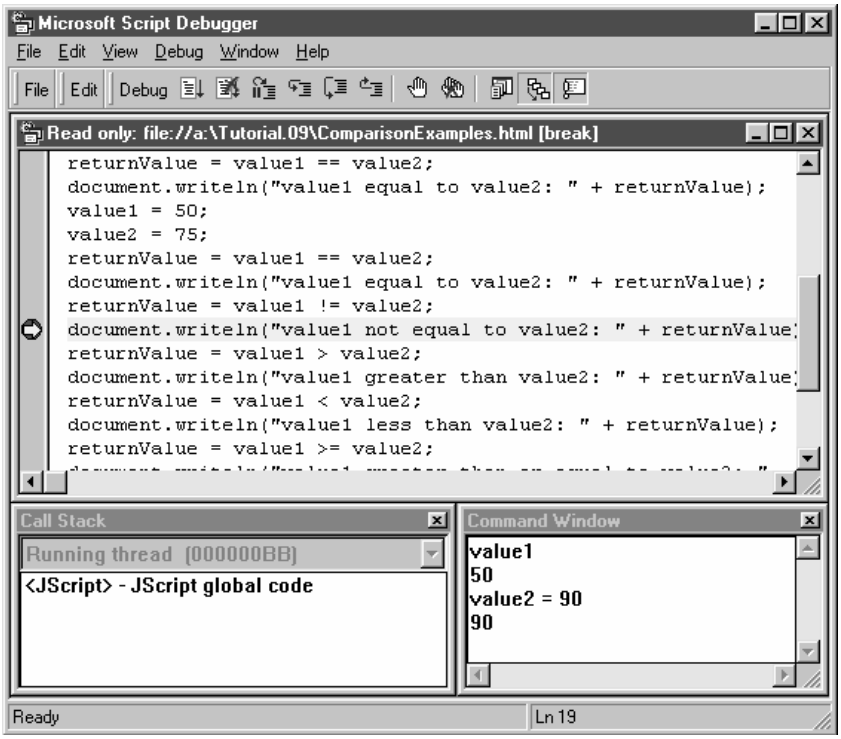


图 9-23　在命令窗口中查看和改变值

4．选中 Debug 菜单下的 Run 执行代码直到遇到第一个断点，接着选中 Debug 菜单下的 Step Into 命令执行 personnel = 50;语句。

5．选中 View 菜单下的命令窗口显示命令窗口。

6．将输入光标移至命令窗口并输入 personnel，接着按下回车键。将在命令窗口内打印 personnel 变量的值 50。

7．下一步，选中 Debug 菜单下的 Run。程序接着运行并在第二个断点处停止下来。选中 Debug 菜单下的 Step Into 命令执行 personnel = 100;语句。

8．将输入光标移至命令窗口并输入 personnel，接着按下回车键。将在命令窗口内打印 personnel 变量的新值 100。

9．当输入光标仍在命令窗口内时输入 personnel = 200，接着按下回车键。personnel 变量的值将变为 200 并在命令窗口的上面部分打印出来。

10．选中 Run 命令执行程序的剩余部分。

11．选中 File 菜单下的 Exit 关闭脚本调试器。

调用堆栈窗口

脚本调试器的调用堆栈窗口与 JavaScript 调试器的调用堆栈窗口具有相同的功能。在调试程序时它提供了查看调用堆栈内容的功能。

下一步，将单步执行 CompanyObjects.html 内的某些函数来观察调用堆栈窗口的内容。

练习跟踪变量：

1．切换至 Internet Explorer 中的 CompanyObjects.html 文档。

2．选中 View 菜单下脚本调试程序中的 Break at Next Statement，接着单击“ Refresh ”按钮。程序开始执行，然后切换到脚本调试器并在第一条语句处暂停下来。注意<Jscript> -Jscript global 代码被添加到调用堆栈窗口内了。这些文字表示当前调用堆栈内的第一条语句，sales.prototype = new Company;。

3．选中 Debug 菜单下的 Step Into。程序控制转移到 Company()函数，接着在第一个调用之前<Jscript> -Company 被添加到调用堆栈窗口。

4．选中 Debug 菜单下的 Step Out。执行 Company()函数内的剩余部分，程序控制转移到下一条语句，Production.prototype = new Company;，并且<Jscript> -Company 被从调用堆栈中删除掉。再选中 Step Into 命令将再次将 Company()添加到调用堆栈中，因为这次是 Production.prototype = new Company;语句调用函数。

5．选中 Run 执行程序的剩余部分。

6．关闭 Internet Explorer 和脚本调试器。

9.2.5 JavaScript 语句的错误和调试资源

若尝试了所有能够想到的方法来确定程序中的错误仍未能发现错误，那么请考虑下面的可能性：您遇到的可能是 JavaScript 语言自身的众所周知的错误。例如，大家所知的一个错误是 onSelect 事件处理器不能在 Windows 平台下运行。无论怎么尝试，由于 JavaScript 语言的错误，下面的代码都不能在 Windows 平台下运行：

```
<FORM>
<INPUT TYPE="text"onSelect="alert('This alert method does
not execute on Windows platforms');">
</FORM>
```

要查阅 JavaScript 错误的清单，请访问 http://developer.netscape.com/support/bugs/kown/index.html 下的 Netscape 的 JavaScript Knows Bugs 页面。然而，请注意应用程序的厂商并不总是第一个知道它们产品中存在错误的。具有创造力的用户通常首先发现错误，接着报告给应用程序的开发商。通常这些用户还喜欢与其他用户共享发现的错误。利用众多的 JavaScript 程序员（通常他们也愿意）来帮助解决问题或跟踪错误。可以在众多不同的网址上、新闻组中和在特殊的 Internet 服务提供商（如 CompuServe、Prodigy 和 American Online）的讨论组中找到帮助。表 9-1 列出了一些 Internet 调试资源。

表 9-1 中的地址是 2000 年 1 月份的。在访问上述的地址时，请记住 Internet 是不断变化的。正如其他商务一样，网站可能随时间改变它们的地址或是关闭。

表 9-1　Internet 调试资源

资　　源	地　　址
Netscape 的 JavaScript 新闻组（需要订阅 Netscape's developer program）	http://developer.netscape.com/members/doc/subscriber/doc/newsgroups/javascript.html
Netscape 的 JavaScript 常见问题解答（FAQ）	http://developer.netscape.com/support/faqs/index.html?content=champions/javascript.html
Usenet JavaScript 新闻组	comp.lang.javascript
Website Abstraction	http://wsabstract.com/
JavaScript.com	http://www.javascripts.com
JavaScript Source	http://javascript.internet.com/
ZDNet Developer	http://www.zdnet.com/devhead/resources/scriptlibrary/javascript/
WebDeveloper	http://webdeveloper.com/javascript/

9.2.6　总结

- ✧ 可以使用 for…in 循环来确定对象中的属性是否被赋给了正确的值。
- ✧ Navigator 包含了两个方法 watch()和 unwatch()，它们被用来设置和取消特殊对象的查看点。
- ✧ 查看点是一种特殊的属性，在程序执行过程中可以监视它们的变化。
- ✧ watch()方法为对象属性设置查看点。
- ✧ unwatch()方法取消先前声明的查看点。
- ✧ JavaScript 调试器是 Netscape 的用来调试 JavaScript 的工具。
- ✧ 脚本调试器是 Microsoft 的用来调试脚本语言（包括 JavaScript 和 VBScript）的工具。
- ✧ 中断模式暂时挂起（或暂停）程序的执行以便能够监视值和跟踪程序执行。
- ✧ Step Into 命令执行一行的代码，接着暂停下来，直到指示调试器继续执行。
- ✧ Step Over 命令允许 Step Over 函数调用。
- ✧ Step Out 命令执行当前函数内的所有剩余代码。
- ✧ 断点是代码中的一条语句，在此处程序执行进入中断模式。
- ✧ 跟踪程序的执行能够确定值改变的准确位置和确定逻辑错误的原因。
- ✧ 在 JavaScript 调试器的调试过程中，可以在中断模式下的使用控制台窗口、查看窗口和 Evaluate 命令来监视变量和表达式。
- ✧ 过程（如函数、方法或事件处理器）在程序中执行的顺序称为调用堆栈。
- ✧ 在脚本调试器的调试过程中，可以在中断模式下使用命令窗口来监视变量和表达式。

- ✧ JavaScript 调试器和脚本调试器都使用堆栈调用窗口来监视调用堆栈。
- ✧ 若尝试了所有能够想到的方法来确定程序中的错误仍未发现原因，那么请考虑一下下面的可能性：遇到的可能是 JavaScript 语言自身的众所周知的错误。

9.2.7 问题

1．最有效的观察赋给对象属性的值的方法是什么？

a．if…else 语句

b．for…in 语句

c．switch 语句

d．内部的 JavaScript 属性对象

2．在 Navigator 中为执行 changedSlogan()函数的 companyInfo 对象的 slogan 属性添加查看点的正确语法是哪一条？

a．changeSlogan()=companyInfo.watch("slogan");

b．companyInfo.watch("slogan"=changeSlogan();

c．companyInfo.watch("slogan",changeSlogan());

d．companyInfo.watch("slogan",changeSlogan);

3．在 Navigator 中怎样取消查看点？

a．unwatch()方法

b．stop()方法

c．stopwatch()方法

d．取消观察的惟一方法是删除包含了 watch()方法的语句

4．____模式暂时挂器（或暂停）程序的执行以便能够监视值和跟踪程序的执行。

a．中断（Break）

b．停止（Stop）

c．挂起（Suspend）

d．等待（Wait）

5．在 JavaScript 调试器中，____命令指示 JavaScript 调试器在遇到 JavaScript 代码后立即进入中断模式。

a．停止执行（Stop Execution）

b．设置中断（Set Interrupt）

c．在下一条语句中断（Break at Next Statement）

d．暂停程序（Pause Program）

6．哪一条调试器命令执行下一行代码？

a．Step Into

b．Step Out

c．Step Over

d．Continue

7．哪一条调试命令执行下一个函数内的所有语句？

a．Step Into

b．Step Out

c．Step Over

d．Continue

8．哪一条调试命令执行函数内的所有剩余的语句并将控制转移到调用当前函数的语句后的下一条语句？

a．Step Into

b．Step Out

c．Step Over

d．Continue

9．哪一条调试命令继续正常地执行程序？

a． Continue

b． Proceed

c． Exit Debug

d． Run

10．____是一条语句，在此处程序执行进入中断模式。

a．停止标记

b．断点

c．暂停位置

d．中断

11．JavaScript 调试器可以使用____窗口来监视变量和表达式。

a．控制台

b．命令

c．值

d．表达式

12．脚本调试器可以使用____窗口来监视变量和表达式。

a．控制台

b．命令

c．值

d．表达式

13．过程（如函数、方法或事件处理器）在程序中执行的顺序是____。

a．执行链

b．过程堆

c．调用堆栈

d．方法批处理

14．下面的哪一种事件处理器不能在 Windows 平台下运行？

a．onChange

b．onFocus

c．onClick

d．onSelect

9.2.8 练习

1．开发有效的程序的最重要的一个方面是项目的设计和分析阶段。一个不错的设计和分析阶段是减少程序中的错误的关键。在 Internet 或本地库中搜索关于此主题的内容。接着撰写一页关于处理软件项目设计和分析阶段方法的论文。

2．在程序开发过程中同等重要的是测试阶段。在 Internet 或本地库中搜索关于程序测试的内容。接着设计一个方案用于在 Web 上发布 JavaScript 程序之前完全测试它们。

3．访问 http://developer.netscape.com/support/bugs/known/index.html 下的 Netscape 的错误清单。研究当前已知的 JavaScript 编程语言中的不同类型的错误。撰写不同类型的错误的分析，内容包括错误的类别、错误所影响的平台和浏览器以及每种错误出现的工作区。

第 10 章　服务器端 JavaScript

案例

WebAdventure 想记录访问者访问他们网址的次数和显示包含客户清单的“客户簿”，还要让网址访问者能够在客户簿上签名。在研究这些特性之后，发现使用客户端 JavaScript 无法记录此信息，因为惟一所知的状态管理方法是将信息存储在每个用户本地计算机上的 Cookies 中。为了能够维护所需的跨不同用户会话的信息，必须使用服务器端 JavaScript。在使用服务器端 JavaScript 之前，必须了解一点关于客户/服务器处理过程的相关知识，以及不同类型服务器端 JavaScript 之间的区别。

浏览 WebAdventure 主页

在本教程中将创建能够作为 WebAdventure 网址使用的程序。程序将保存它所接收到的点击数，并包含了一块访问者能够在其上签署客户簿的区域。所创建的程序并不是一个完整的、强壮的网址，因为它仅仅含有点击计数器和客户簿。

本章的 10.1 节介绍服务器端 JavaScript——Netscape 的 LiveWire。10.2 节介绍服务器端 JavaScript——Microsoft 的 Active Server Pages（ASP）。为了浏览 WebAdventure 主页程序，必须能够访问 Netscape 或 Microsoft Web 服务器，每节都包含了从何处下载每个公司的服务器应用程序的信息。浏览使用了截获的、正在 Internet Explorer 中运行的 Microsoft 的 ASP 图像来演示此程序。然而，请注意尽管程序显示的结果和执行的功能与在 Navigator 中的相同，但是 ASP 和 LiveWire 的重要代码有点不同。

提示：若含有 Data Disk 的驱动器不在 Netscape 或 Microsoft Web 服务器的控制之下，那么将不能运行 Data Disk 的 Tutorial.10 文件夹内的程序。

浏览 WebAdventure 主页：

1．WebAdventure 的 ASP 程序的首页为 Tutorial10_StartPage.asp。图 10-1 显示了在 Internet Explorer 中运行的 Tutorial10_StartPage.asp。

2．当用户在文本框内输入他的或她的姓名并单击“Continue”按钮之后，将打开 Tutorial10_HomePage.asp 文档，如图 10-2 所示。注意在第一段中就使用了在 Tutorial10_StartPage.asp 内所输入的姓名。此页还包括了显示此页已被访问次数的点击计数器，以及签名或查看客户簿的按钮，后面将会看到每个按钮所调用的文档实例。

3．在文本编辑器或 HTML 编辑器中，从 Data Disk 的 Tutorial.10 文件夹内打开 Tutorial10_HomePage.asp 并检查 JavaScript 代码。请注意包含了不熟悉的代码语法<%…%>标签对。这些标签对和语法是服务器端 JavaScript 代码的 ASP 语法。关闭 Tutorial10_HomePage.asp 文件。

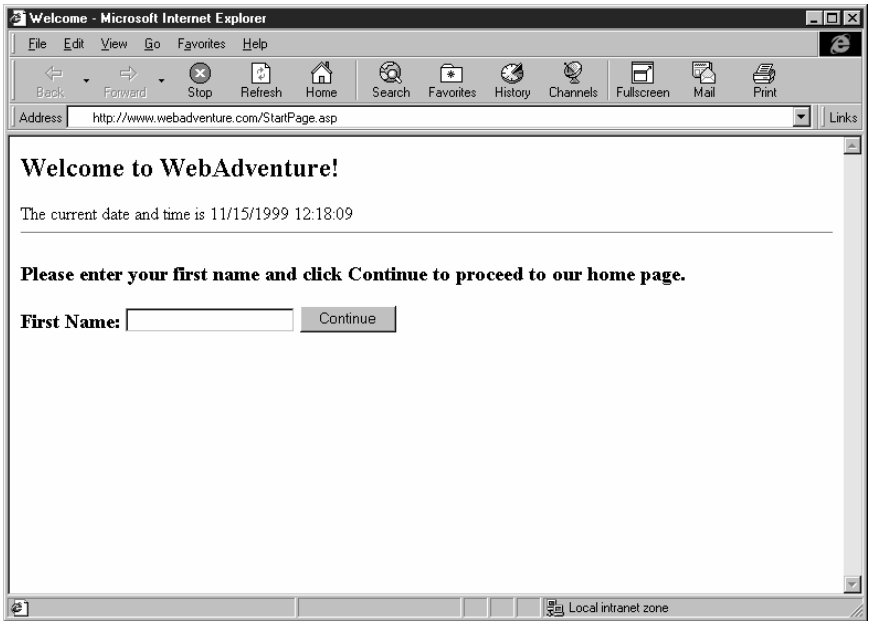


图 10-1　Internet Explorer 中的 Tutorial10_StartPage.asp

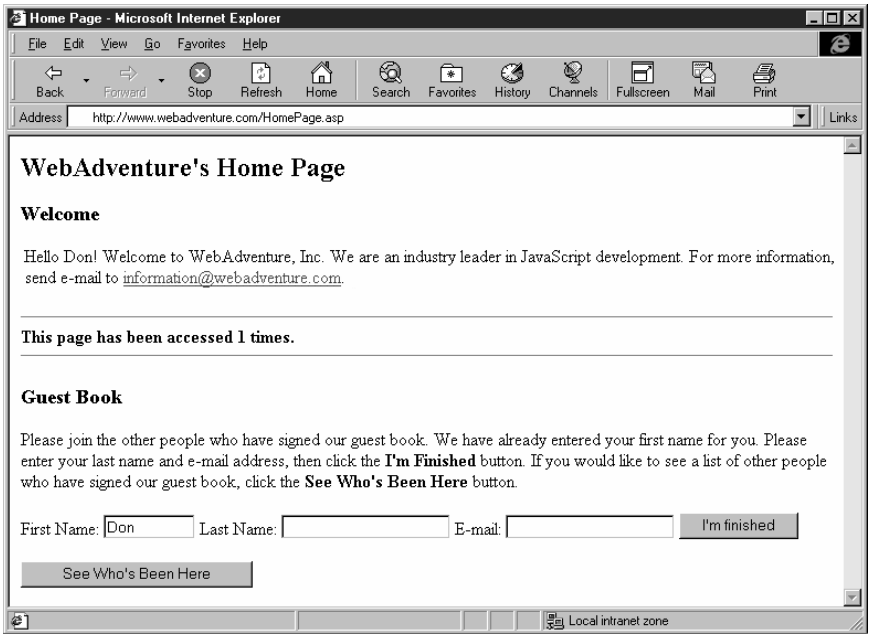


图 10-2　Internet Explorer 中的 Tutorial10_HomePage.asp

4．下一步在文本编辑器或 HTML 编辑器中，从 Data Disk 的 Tutorial.10 文件夹内打开 Tutorial10_HomePage.html 文件并检查 JavaScript 代码。Tutorial10_HomePage.html 是 LiveWire 版的应用程序。LiveWire 使用<SERVER>…</SERVER>标签对和“`”字符来指明服务器端 JavaScript 代码。

提示：ASP 应用程序使用.asp 文件后缀，而 LiveWire 应用程序使用.html 后缀。

5．关闭文本编辑器或 HTML 编辑器。

10.1　Netscape LiveWire

本节目标

在本节将学习：

- ✧ 客户/服务器结构
- ✧ 开发服务器端 JavaScript
- ✧ 怎样创建 LiveWire 应用程序
- ✧ LiveWire 核心对象
- ✧ 怎样使用 LiveWire 创建客户簿

10.1.1　客户/服务器结构

在 Web 的客户/服务器环境中 Web 浏览器是客户。直到现在，所关注的是在 Web 浏览器中开发的客户端 JavaScript。为了学会完整的 Web 的开发技能，还需要理解 Web 的服务器端——服务器端 JavaScript 是怎样适应 Web 开发的。在深入理解服务器端 JavaScript 和开始本节的学习之前，讨论一下客户/服务器结构的基础知识。

存在多种对客户/服务器系统的定义。在传统的客户/服务器结构中，服务器通常是某种类型的数据库，客户从其中请求信息。服务器通过维护请求或将所请求的信息提供给客户来完成对信息的请求——术语客户/服务器由此而来。由客户和服务器组成的系统就是众所周知的两层系统（Two-tier System）。在两层系统中客户或前端的主要作用之一是为用户提供接口。用户接口从用户那里收集信息，并将它提交给服务器或后端，接着接收、格式化和展示服务器返回的结果。服务器的主要任务是存储和维护数据。在客户/服务器系统上的大量的处理（如计算）通常发生在服务器上。然而，随着桌面计算机功能日益强大，许多客户/服务器系统至少将某些处理任务交给客户来完成。在典型的客户/服务器系统中，客户计算机包含了用来从服务器数据库中收集信息的前端。服务器定位满足客户请求的记录，接着将信息返回给客户。客户计算机可能还执行某些处理如构造发送给服务器的查询或格

式化和展示返回的数据。图 10-3 演示了两层客户/服务器系统的设计。

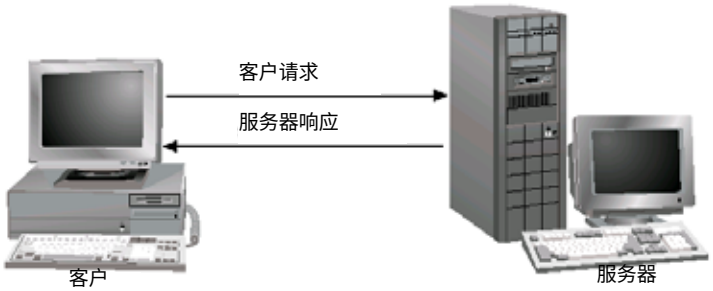


图 10-3　两层客户/服务器系统设计

提示：术语分布式应用程序用来描述多个计算机为一个应用程序分担计算任务，因此处理是“ 分布的 ”。

Web 构建在两层客户/服务器系统上——Web 浏览器（客户）从 Web 服务器请求 HTML 文档。Web 浏览器是客户用户接口，并且 Web 服务器可以看作是“ Web 页数据库 ”（从某种意义上说）。在 Web 服务器返回所请求的 HTML 文档之后，Web 浏览器（作为客户用户接口）立即负责将文档格式化和展示给用户。

在为 Web 服务器安装数据库和其他类型的应用程序之后，客户/服务器系统立即演化为众所周知的三层客户结构。三层、多层或客户/服务器系统由三个明显的部分组成：客户层、处理层和数据存储层。客户层或用户接口层还是 Web 浏览器。然而，两层客户/服务器系统中的数据库部分被分为处理层和数据存储层。处理层或中间层是一个“ 处理桥梁 ”，它处理 Web 浏览器客户和数据存储层之间的交互。实质上，客户层请求服务器上的数据库。处理层根据来自客户层的请求执行任何必需的处理或计算，接着从数据存储层读取或写入信息。处理层还将任何信息返回给客户层。注意处理层并不是惟一的执行处理的层。仍是 Web 浏览器（客户层）显示 HTML 文档（它需要处理），而数据存储层中的数据库或应用程序可能还执行必需的处理。客户端 JavaScript 在客户层使用，而服务器端 JavaScript 在处理层使用。图 10-4 演示了三层客户/服务器系统的设计。

提示：两层客户/服务器接口是一个物理布局，其中客户和服务器是两台单独的计算机。而三层客户/服务器架构比布局更概念化，因为处理层和数据存储层通常位于相同的计算机上。

设计客户/服务器系统的一个重要方面是确定客户执行的多重处理。对开发 Web 网站来说，必须确定是否使用客户端或服务器端 JavaScript。此选择听起来让人困惑，因为客户端 JavaScript 和服务器端 JavaScript 共享大部分相同的核心来执行大量的计算和数据存储。在处理 Web 时任务的分工特别重要。与使用私用网络不一样，不知道 Web 上的每个客户的处理能力。不能假设访问客户/服务器应用程序（网址）的每个客户都能够执行应用程序所需的处理。因此，大量的处理应交给服务器执行。

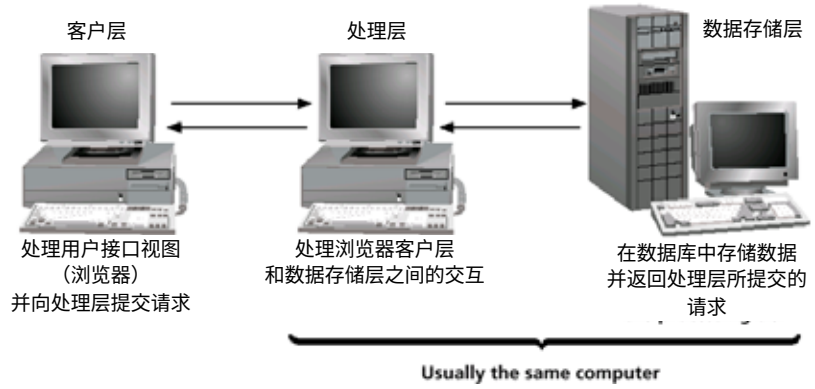


图 10-4　三层客户/服务器系统设计

既然服务器通常比客户计算机的功能更强，那么首先想到的可能是让服务器执行所有的处理而仅使用客户机来显示用户接口。尽管不想让客户机频繁地执行它们所不能处理的的任务，但是存在多种理由让客户机尽可能多地执行处理。首先，在多个客户机之间分布处理将创建功能更强大的应用程序，因为不再受单个计算机处理能力的限制。每隔一天客户计算机功能就变得更强，并且在本地 Web 浏览器也能够使用先进的功能如 JavaScript 和 DHTML。第二，客户计算机的本地处理减少了通过 Internet 传输的时间，并使应用程序运行得更快。若客户不得不等待服务器上执行完所有的处理，那么繁忙的 Internet 连接上 Web 应用程序将会是多么的缓慢。最后，在客户计算机上执行处理将减轻在服务器上的处理负载。若三层客户/服务器系统中的所有处理都放在服务器上，那么受欢迎网址的、传输量大的服务器将试图处理来自大量客户的请求，这时它的任务就十分繁重了。

10.1.2　开发服务器端 JavaScript

在第 6 章讨论表单和 CGI 时，接触到了三层客户/服务器处理。将表单数据提交给处理层的服务器之前，客户层使用 JavaScript 来验证表单数据。在处理层，CGI 以及脚本语言如 Perl 通常在将数据提交给数据存储层之前以某种方式对数据进行准备和处理。在发明服务器端 JavaScript 之前，CGI 是开发三层客户/服务器应用程序最常用的方法之一。然而，CGI 不是真正的编程语言。相反，CGI 是一种通常与脚本语言如 Perl 一起使用的协议，用来处理客户层和数据存储层之间的通信。

相比之下，服务器端 JavaScript 是一种基于客户端 JavaScript 的编程语言，它能够处理客户层和数据存储层之间的通信。另外，服务器端 JavaScript 能够紧密地与客户端 JavaScript 交互，因为它们具有相同的基本编程特性。因此，许多开发人员喜欢使用服务器端 JavaScript 来处理三层客户/服务器系统的处理层。

然而，服务器端 JavaScript 并不是没有缺点。此语言最大的缺点之一是它具有产权并且因不同厂商而异，必须理解每个厂商 Web 服务器的服务器端 JavaScript。它不存在类似于 ECMAScript 的服务器端标准。因为缺乏标准，所以 CGI 仍然是处理层常用的技术，很

显然每种类型的服务器都支持它。

尽管服务器端 JavaScript 具有产权并且因不同厂商而异，但是大部分实现使用十分相同的核心语法（在客户端 JavaScript 已经学过）。能够使用在客户端 JavaScript 中所学习的技能来编写服务器端 JavaScript 程序，不用理会使用何种版本的服务器端 JavaScript。

用服务器端 JavaScript 开发的 Web 应用程序不需要数据存储层。例如，能够创建一个服务器端 JavaScript 程序，它惟一目的是动态创建给客户显示的 Web 页面。另一个不需要数据存储层服务器端 JavaScript 程序是计算 Web 网址的点击数。不含有数据存储层的 Web 应用程序不是三层客户/服务器系统，我们称这些应用程序为分布式应用程序。在本节内创建的服务器端 JavaScript 程序是不含有数据存储层的分布式应用程序。

提示：在第 11 章中，将继续使用服务器端 JavaScript，还将学习怎样创建使用数据库的三层客户/服务器系统。

如上所述，服务器端 JavaScript 的最大障碍之一是它具有产权并且因不同厂商而异。因为本教程重点讨论 Netscape 和 Microsoft 技术，所以在本教程中将讨论它们的服务器端 JavaScript。10.1 节讨论了 Netscape 的服务器端 JavaScript。在 10.2 节，将学习 Microsoft 的在 ASP（Active Server Pages）中使用服务器端 JavaScript。

为了实现本节的练习，必须使用能够支持 LiveWire 的 Netscape Enterprise 或 FastTrack 服务器。本教程的实例和图像是使用 Netscape Enterprise Server Standard Edition 3.6 创建的。若正在使用 FastTrack 服务器，那么实现本教程练习所需的步骤可能与使用 Enterprise 服务器所需的过程不同。可以从 http://developer.netscape.com/下载 Netscape Web 服务器程序。在下载服务器程序之前，确保有足够的磁盘空间。若 Internet 连接比较慢，那么下载时要花费好多的时间，因为 Enterprise 服务器程序由 36MB 的压缩文件组成。另外，在下载程序之前，请阅读每种服务器的在线数据来确定计算机是否能够满足最小硬件需求。

帮助：安装 Web 服务器可能是个复杂的过程，它需要深入了解网络技术和传输协议。另外，还必须理解网络管理功能来正确地维护 Web 服务器。既然本教程的目的是介绍 JavaScript，那么将不在安装服务器或管理过程上花费时间。若您自学，那么请参阅 Netscape 和 Microsoft 网址来学习有关安装和管理本教程内所提到的 Web 服务器的详细信息。若正在参加培训，那么请向教师请教怎样使用企业内服务器的特殊用法。

提示：以前可以将 LiveWire Server Extension 作为独立的产品（LiveWire 1.0）使用，但是现在它被集成到 Netscape Web servers version 3 和更高版本中去了。若使用的 Netscape 服务器版本比 3 低，那么将需要取得 LiveWire Server Extension（LiveWire 1.0）。

提示：本教程仅简要地介绍了 LiveWire 和服务器端 JavaScript。要了解更多的与 LiveWire 和服务器端 JavaScript 相关的内容，请访问 Netscape 开发人员网址 DevEdge Online，http://developer.netscape.com/的 Document 部分。还能从在线帮助发现关于每种 LiveWire 开发环境组件的优秀文档。

10.1.3　创建 LiveWire 应用程序

在深入服务器端 JavaScript 语言细节之前，需要理解怎样创建服务器端 JavaScript 应用程序。与客户端 JavaScript 不同，在使用 JavaScript 之前，必须编译和安装 LiveWire 应用程序。创建服务器端 JavaScript 应用程序的特殊步骤如下：

1．创建服务器端脚本。

2．使用 jsac 编译器编译和发布应用程序。

3．使用 JavaScript 应用程序管理器安装和启动应用程序。

让我们详细地观察每个步骤。

服务器端脚本

在 HTML 文档或.js 源文件中创建服务器端 JavaScript 脚本，这与创建客户端 JavaScript 一样。然而，当客户从服务器请求 HTML 文档时，在将文档提交给客户之前服务器先执行所有服务器端 JavaScript。在 Web 浏览器接收完文档之后，它立即执行客户端脚本。

HTML 文档中含有的服务器端 JavaScript 封装在<SERVER>…</SERVER>标签对内，与客户端 JavaScript 相似。下面的代码包含了一个<SERVER>…</SERVER>标签对实例，其中只封装了一个 write()方法。服务器端 JavaScript 的 write()方法与客户端的 write()方法执行十分相同的功能，除了客户端 write()是将文本输出给浏览器而服务器端 write()是将信息返回给客户。

```
<SERVER>
write("Hello World");
</SERVER>
```

注意在上面的代码中 write()方法不含有 Document 对象，而客户端 JavaScript 的 write()方法含有 Document 对象。服务器端 JavaScript 中的 write()方法是全局的，它并不与任何特殊的对象关联在一起。实际上，服务器端 JavaScript 并不引用 Document 对象、Window 对象或其他众多的与浏览器相关的客户端 JavaScript 对象。相反，服务器端 JavaScript 仅涉及处理层那部分的对象（在本节的后面部分将学习服务器端 JavaScript 对象）。

提示：请查阅 Netscape 的服务器端 JavaScript 文档有关全局方法的清单。

服务器端 JavaScript 保存在具有.js 后缀的源文件中，客户端 JavaScript 源文件也如此。服务器端源文件对维护函数库十分有用，函数库能够被应用程序中的所有服务器端 JavaScript 代码访问。与客户端 JavaScript 不一样，不能使用 src 属性来引用服务器端源文件的名称。但是，在编译应用程序时，可以包含.js 源文件名。在编译的过程中，编译器让应用程序中的其他 HTML 文档使用.js 源文件中的 JavaScript 代码。

若需要在 HTML 标签中包括服务器端 JavaScript，那么请用反向引号封装代码。请使用键盘上的“ ` ”字符或转义字符“ \Q ”来表示反向引号（通常在数字键 1 的左边）。例如，

服务器端 JavaScript 编程中的 Client 对象被用来创建自定义的客户属性。假设在 Client 对象中已创建了一个自定义属性 email，并且想在使用 mailto:属性的链接中使用此属性。要返回服务器端 JavaScript Client 对象的自定义 email 属性并在链接中使用它，那么请使用与下面相似的语法：

```
<A HREF="mailto:" + `client.email`>
Click here to send an e-mail</A>
```

另外，可以使用如下转义字符：

```
<A HREF="mailto:" +    \Qclient.email \Q>
Click here to send an e-mail</A>
```

在 Netscape 服务器将含有服务器端 JavaScript 的 HTML 文档传递给客户之前，它执行所有<SERVER>…</SERVER>标签对内的内容。若客户想在它接收到 HTML 文档之后查看源文档，那么将看不到<SERVER>…</SERVER>标签对和它们所封装的代码。但是，客户仅能看到代码返回的结果。例如，图 10-5 显示了含有客户端 JavaScript 的 HTML 文档以及用<SERVER>…</SERVER>标签对和反向引号封装的服务器端 JavaScript。图 10-6 显示了客户端上具有客户端和服务器端 JavaScript 的 HTML 文档内容，实例假设 Client 对象的 email 属性值是 dongosselin@compuserve.com。

```
<HTML>
<HEAD>
<TITLE>Client and Server-Side JavaScript</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
document.write("<H2>This line is generated by client-side
JavaScript.</H2>");
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
<BODY>
<SERVER>
write("<H2>This line is generated by server-side
JavaScript.</H2>");
</SERVER>
<A HREF="mailto:" +    \Qclient.email \Q>
Click here to send an e-mail to Don Gosselin</A>
</BODY>
</HTML>
```

图 10-5　包含客户端和服务器端 JavaScript 的 HTML 文档

```
<HTML>
<HEAD>
<TITLE>Client and Server-Side JavaScript</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
document.write("This line is generated by client-side JavaScript.");
document.write("<BR>");
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
<BODY>
<H1>This line is generated by server-side JavaScript.</H1>
<A HREF="mailto:dongosselin@compuserve.com" >
Click here to send an e-mail to Don Gosselin</A>
</BODY>
</HTML>
```

图 10-6　客户端上具有客户端和服务器端 JavaScript 的 HTML 文档

下一步,将创建一个简单的文档,作为在浏览器所看到的 WebAdventure 主页的起始页：

1．启动文本编辑器或 HTML 编辑器，并创建新文档。

2．输入文档的起始<HTML>、<HEAD>、<TITLE>和<BODY>段，以及标题标签。

```
<HTML>
<HEAD>
<TITLE>Welcome</TITLE>
</HEAD>
<BODY>
<H2>Welcome to WebAdventure!</H2>
```

3．输入下面的<SERVER>…</SERVER>标签对，它将创建 Date 对象，接着使用 toLocaleString()方法将日期传送给客户。

```
<SERVER>
var curDate = new Date();
write("The current date and time are " +
curDate.toLocaleString());
write("<HR>");
</SERVER>
```

4. 添加下面的标题标签、含有用来输入用户名的文本域和提交按钮（用来调用主 HTML 文档 HomePage.html）的表单。程序使用提交按钮来将文本域内的值提交给

HomePage.html 文档。

```
<H3>Please enter your first name and click Continue
to proceed to our home page.</H3>
<FORM METHOD="post" ACTION="HomePage.html">
First Name: <INPUT TYPE="text" NAME="first">
<INPUT TYPE="submit" VALUE=" Continue ">
</FORM>
```

5．添加下面的代码结束<BODY>和<HTML>标签。

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

6．在 Data Disk 的 Tutorial.10 文件夹内将文件存为 StartPage.html。在作为 LiveWire 应用程序执行之前，需要编译和安装此程序。

编译和发布应用程序

在创建了服务器端脚本（HTML 文档和.js 源文件）之后，必须进行编译。在成功编译完应用程序之后，立即使用 jsac 程序编译 LiveWire 应用程序，将.web 文件拷贝到 Web 服务器上。jsac 是用来将服务器端脚本编译为一个带有.web 后缀的文件的应用程序。Netscape 服务器将.web 后缀作为可执行的应用程序。被编译为一个.web 文件的相关文件的集合被称为 LiveWire 应用程序。尽管组成 LiveWire 应用程序的文件被存储在.web 后缀的文件中，但是用户可以通过在 Web 浏览器客户端中输入文件的完整名和.html 后缀来访问它们。可以用多种不同的选项组合来执行 jsac 应用程序，并且在命令行输入 jsac –h 来查看编译器选项清单。

要创建含有多个应用程序的 LiveWire 应用程序，请在 jsac 命令的后面添加作为程序一部分的每一个文件名，它们之间用空格分割。下面的代码显示了怎样使用 jsac 应用程序来编译由两个文件组成的服务器端 JavaScript 应用程序：homepage.html 和 functionlibrary.js。程序使用了两个选项：-v 用来显示编译过程的具体描述，-o 用来创建名为 main.web 的.web 文件。

```
jsac –v –o main.web homepage.html functionlibrary.js
```

下一步，编译 StartPage.html：

1．切换到系统命令提示下。每个操作系统使用的命令提示不同。在 Windows NT 下使用命令提示，请选择 Start 菜单下的 Run 并输入 cmd。

2．将目录改变为 Data Disk 的 Tutorial.10 文件夹。

3．输入下面的命令编译应用程序，接着按下回车键。默认情况下，jsac.exe 应用程序安装在 LiveWire 服务器上的 c:\Netscape\SuiteSpot\bin\https 文件夹内。若 jsac.exe 安装在系

统上的其他驱动器和文件夹内，那么请用相应的路径代替 c:\ Netscape\SuiteSpot\bin\https。

```
c:\Netscape \SuiteSpot \bin \https \jsac -v -o
WebAdventure.web startpage.html
```

帮助：因为空间的限制，上面的命令含有一个换行符。请在按下回车之前确保将整条命令输在一行内。

4．在程序编译完后，请关闭命令提示窗口。

安装和启动应用程序

应用程序管理器安装和管理 Web 服务器上的 LiveWire 应用程序。在成功编译完应用程序之后，请使用 URL http://server.domain/appmgr/ 来打开应用程序管理器，用服务器名和域名代替上面的 server 和 domain。应用程序管理器是一个 JavaScript 应用程序，将在浏览器中打开它。应用程序管理器的左帧包含了一个表单，其中含有已安装应用程序的清单和一列执行各种命令的链接。在顶帧是几个命令按钮，其中包括了一个“ Add Application ”按钮，可以用它来安装和启动 LiveWire 应用程序。单击左帧或顶帧内的链接将会在右帧内显示相关的页面。图 10-7 显示了在单击“ Add Application ”按钮后，应用程序管理器内的 Add Application 表单实例。

图 10-7　应用程序管理器的 Add Application 表单

Add Application 表单被用来将应用程序添加到 Web 服务器。表 10-1 描述了每个表单域的作用。

表 10-1 Add Application 表单域

域	描 述
Name	应用程序名。不要为应用程序取一个与 Web 服务器上已有的目录名相同的名称，否则客户将不能够访问该目录下的文档
Web File Path	使用 jsac 命令创建的应用程序.web 文件的完整路径名
Default Page	在客户请求未指定文件时，传递给客户的默认页。此域可选
Initial Page	在首次启动应用程序时执行的第一个页面。此页被用来执行应用程序启动时所需的任何任务，如初始化计数器值或建立数据库连接。此域可选
Built-in Maximum Database Connections	数据库服务器许可所允许的最大数据库服务器连接数目。此域的默认值为 0
External Libraries	与应用程序一起使用的外部库的路径名。此域可选
Client Object Maintenance	用来维护状态信息的技术。可以选择 client-cookie、client-URL、server-IP、server-cookie 或 server-URL

在填充了 Add Application 表单内的所必需的域之后，单击“ OK ”按钮安装应用程序。在添加应用程序之后，将立即自动启动它。接着可以在左帧内的已安装应用程序清单中选中并单击 Run 链接（或在 Navigator 中打开它的 URL）运行此应用程序。

提示：若对应用程序作出任何修改，那么接着必须在应用程序管理器中重新启动应用程序让改动起作用。

下一步，安装和运行 WebAdventureHome 应用程序：

1．启动应用程序管理器，接着单击顶帧内的“ Add Application ”按钮。

2．在 Add Application 帧内的 Name 域中，输入 WebAdventureHome。接着在 Web File Path 域内输入 a:\Tutorial.10\WebAdventure.web（假设 Data Disk 在驱动器 a:内）。

3．在 Default Page 域内输入 StartPage.html。

4．让 Add Application 表单内的其他域保持默认值，接着单击“ OK ”按钮。

5．若需要，从左帧内的已安装应用程序清单中选中 WebAdventureHome，并单击“ Run ”在 Web 浏览器中打开 StartPage.html 文件。图 10-8 显示了在 Navigator 中的该文档实例。

6．在 Navigator 中打开 StartPage.html 之后，选中 View 菜单下的 Page Source。在 Page Source 窗口中，请注意文档内未包含服务器端 JavaScript 的 write()语句。相反，客户仅接收到了语句的输出结果。

7．关闭 Page Source 和 Navigator 窗口。

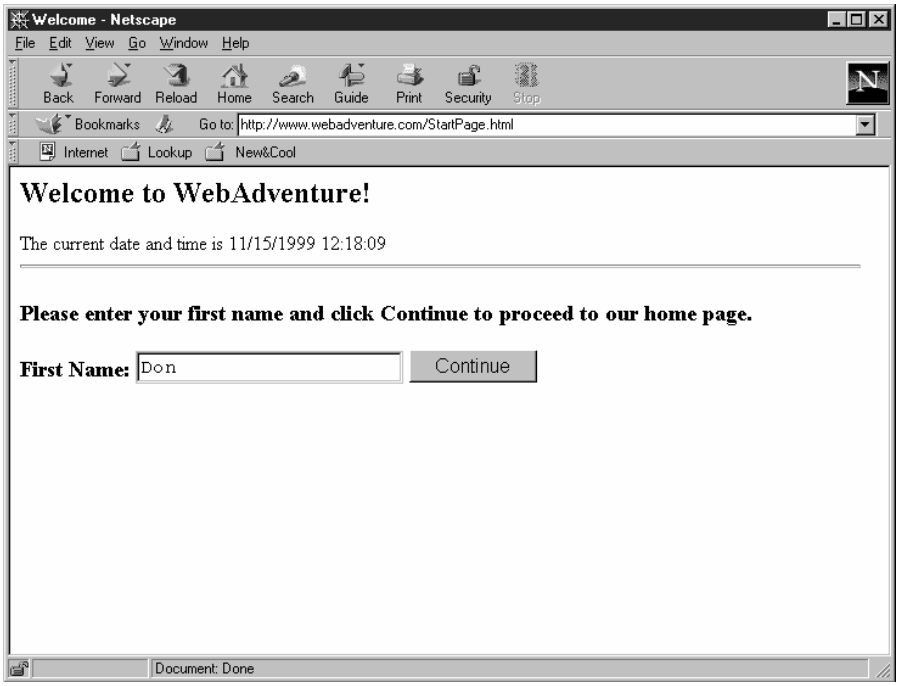


图 10-8　Navigator 内的 StartPage.html

10.1.4　LiveWire 核心对象

LiveWire 不能像客户端 JavaScript 那样能够识别浏览器的 Document 和 Window 对象。相反，LiveWire 能够识别四种内嵌的对象：Request、Client、Project 和 Server，它们在客户层和数据存储层（若存在一个）之间的处理层运行。Request、Client、Project 和 Server 对象在 LiveWire 中称为会话管理对象。尽管会话管理对象内嵌在服务器端 JavaScript 中，但是它们每个具有不同的生存期和可用性。可以使用每个对象来访问处理层内的特定信息和在服务器上存储状态信息。现在来详细讨论每种对象。

提示：可以使用 LiveWire 的 File 对象来永久地记录存储在会话管理对象内的信息。请查阅 Netscape 的服务器端 JavaScript 文档来了解更多的细节。

Request 对象

Request 对象表示来自客户的当前 URL 请求。每次客户请求一个 URL，LiveWire 都将创建一个新的 Request 对象。例如，若用户在浏览器中单击某个链接或选中一个新的 URL，那么 LiveWire 将创建一个 Request 对象。在客户端 JavaScript 使用 document.location 或 history.go()方法时，或在服务器端 JavaScript 使用内嵌的 redirect()方法时，LiveWire 都会创建一个 Request 对象。

提示：redirect()方法用来向客户传递另外一个页面，与使用客户端 JavaScript 的 Location 对象的 href 属性一样。

在所有的会话管理对象中，Request 对象的生存期最短，因为它仅在当前请求未完成时存在。换句话说，在 LiveWire 将 Web 页面传递给客户之后，Request 对象将不存在了。Request 对象包含了几个预定义的属性，它们返回与客户请求相关的信息。表 10-2 列出了 Request 对象的几个常用预定义属性。

表 10-2 Request 对象的预定义属性

属　性	描　述
agent	客户浏览器名和版本号
ip	客户 IP 地址
method	请求的 HTTP 方法
protocol	客户浏览器所支持的 HTTP 协议
imageX	在用户单击图像映射之后，光标的水平位置
imageY	在用户单击图像映射之后，光标的垂直位置

一个重要和有用的属性是 agent 属性，它返回发出 URL 请求的浏览器名和版本号。可以使用此属性根据浏览器的类型来动态生成 HTML 和 JavaScript 代码。例如，若使用 DHTML 为 Web 页增添动画，那么可以为 Navigator 或 Internet Explorer 构建相应的代码，接着将它返回给请求器。它是在第 7 章所学的跨浏览器兼容技术的另一种可选的方法。

提示：可以使用 request.property = value;语法为 Request 对象创建自己的属性。然而，因为此属性仅在 Request 对象存在的时候才存在，所以通常使用 JavaScript 变量更简单些。

Request 对象对检索客户的相关信息十分有用。然而，Request 对象的最有用的特性之一是客户浏览器表单内所有命名元素都被作为 Request 对象的属性添加。回顾一下在单击提交按钮时，表单上的每个域都以“名称=值”对的形式提交给服务器。若没有 LiveWire，那么为了使用“名称=值”对，必须使用 String 对象的方法来解析它们。相比之下，LiveWire 使用元素的 name 属性来创建 Request 对象的属性，并将包含在元素域内的值赋给每个属性。考虑一下在图 10-9 中显示的一个典型的表单。

在将图 10-9 中的表单提交给 LiveWire 文档 ProcessOrder.html 之前，将会创建一个含有代表每个表单元素的属性的 Request 对象。使用下面的属性来引用 Request 对象中的每个表单元素：

```
request.name
request.address
request.city
request.state
request.zip
request.email
```

```
<HTML>
<HEAD>
<TITLE>Customer Information</TITLE>
</HEAD>
<BODY>
<H2>Customer Information</H2>
<FORM METHOD="post" ACTION="ProcessOrder.html"
NAME="customer_information">
Name<BR>
<INPUT TYPE="text" NAME="name" SIZE=50><BR>
Address<BR>
<INPUT TYPE="text" NAME="address" SIZE=50><BR>
City, State, Zip<BR>
<INPUT TYPE="text" NAME="city" SIZE=38>
<INPUT TYPE="text" NAME="state" SIZE=2 MAXLENGTH=2>
<INPUT TYPE="text" NAME="zip" SIZE=5 MAXLENGTH=5><BR>
E-Mail<BR>
<INPUT TYPE="text" NAME="email" SIZE=50><P>
<INPUT TYPE="reset">
<INPUT TYPE="submit">
</BODY>
</HTML>
```

图 10-9　一个典型的表单

任何被作为查询串添加到 URL 的“名称=值”对也被作为 Request 对象的属性添加。下面的代码将查询串添加给一个 URL。假设此 URL 是 LiveWire 应用程序的一部分。

```
<A HREF="http://www.URL.com/TargetPage.html?firstName=Don
&lastName=Gosselin&occupation=writer">Link Text</A>
```

在单击此链接之前，可将 LiveWire 中 Request 对象的 firstName、lastName 和 occupation 属性作为 Request 对象的属性引用。

```
request.firstName
request.lastName
request.occupation
```

表单域和查询字符串“名称=值”对通常被服务器端 JavaScript 程序提交给数据处理层的数据库或其他类型的应用程序。还可以用其他方式使用服务器端 JavaScript 来计算或维护信息，接着将结果返回给客户。

下一步，将创建 WebAdventure 的主 HTML 文档，其中含有对 Request 对象的表单属

性的引用。

创建 WebAdventure 的主 HTML 文档并引用 Request 对象：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．输入文档的起始<HTML>、<HEAD>、<TITLE>和<BODY>段。

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

3．添加下面的标题标签。

```
<H2>WebAdventure ’s Home Page</H2>
<H3>Welcome</H3>
```

4．输入下面段落，它使用 Request 对象插入来自 StartPage.html 文件内名文本域的值。

```
<P>Hello <SERVER>write(request.first);</SERVER>!
Welcome to WebAdventure, Inc. We are an industry
leader in JavaScript development. For more information,
send e-mail to <A HREF="mailto:
information@webadventure.com">
information@webadventure.com</A>.
```

5．添加下面的代码来结束<BODY>和<HTML>标签。

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

6．在 Tutorial.10 文件夹内将文件存为 HomePage.html。使用 jsac 重新编译 WebAdventureHome 程序，请确保在下面的命令行中包括 StartPage.html 文件和 HomePage.html 文件，如下：

```
c:\Netscape \SuiteSpot \bin \https \jsac -v -o
WebAdventure.web startpage.html homepage.html
```

7．使用应用程序管理器重新启动此应用程序。

8．在 Web 浏览器中打开 LiveWire 服务器上的 StartPage.html 文件，接着在 First Name

文本域内输入名并单击“ Continue ”按钮打开 HomePage.html。图 10-10 显示了 HomePage.html 在 Navigator 中的结果。可以看到在 StartPage.html 中输入的名已经被插入到 HomePage.html。

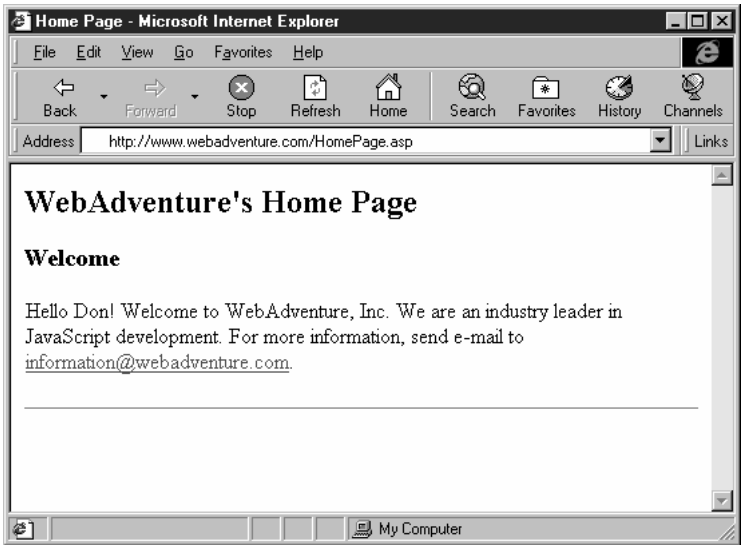


图 10-10　Navigator 中的 HomePage.html

9．关闭 Web 浏览器窗口。

Client 对象

每次客户请求一个 LiveWire URL 时都将初始化一个新的 Request 对象，接着在将 URL 传递给客户之后将立即销毁它。然而，一个 LiveWire 应用程序可能由多个文档组成。为了在 LiveWire 应用程序中跨多个网页保存信息，必须使用 Client 对象。Client 对象用来临时保存 LiveWire 应用程序中所有网页能够访问的具体的客户信息。在客户首次访问指定程序内的 URL 时，将初始化一个 Client 对象。

Client 对象并未包含任何预定义的属性。相反，可以创建自己的 Client 对象属性来保存使用应用程序的用户相关的信息。为 Client 对象创建新属性的语法是 client.property = value;。例如，可以创建包含了用户名的属性。接着能够使用 name 属性来为用户所使用的应用程序内的每个网页创建自定义的欢迎词。Client 对象属性的其他一些用途包括为在线商店存储购物车物品和保存由多部分组成的表单内的域值。假设下面的代码是由多部分组成的表单内的首页。代码显示了服务器端 JavaScript 代码实例，它将表单的域值作为 Client 对象的属性赋值。将代码插入创建由多部分组成表单内的第二个页面的 HTML 文档中。在客户请求第二个页面时执行代码。在实例中，将从 Request 对象相关属性中检索每个表单元素的值。

```
<SERVER>
client.name = request.name; client.address = request.address;
client.city = request.city; client.state = request.state;
client.zip = request.zip; client.email = request.email;
```

```
</SERVER>
```

LiveWire 无法知道客户是否正在与应用程序交互。因为 Web 不像实际网络那样包含任何实际的链接，所以将会出现上述的情况。相反，它依赖 HTTP 在 Internet 上来反复地发送请求和响应。在用户访问网址时，他们仅仅是在请求一个文档。在 LiveWire 服务器返回所请求的文档之后，它就无法知道客户是否还在与应用程序交互。服务器知道客户想继续与应用程序交互的惟一方法是在客户请求另一个文档时。因此，Client 对象的生存期是 10 分钟。可以使用 expiration() 方法来改变生存期。expiration() 方法的语法是 client.expiration(seconds);。注意是以秒来指定时间。若想将 Client 对象的生存期增至为 20 分钟，那么使用 client.expiration(1200);语句，因为 20 分钟就是 1200 秒。

若确定了彻底使用完应用程序，那么就可以使用 destroy()方法来删除 Client 对象。destroy()方法用来删除 Client 对象。例如，假设 LiveWire 应用程序中包含了一个 Complete Sale 按钮用来处理客户订单。在处理完订单之后，就可以立即使用 client.destroy();语句来删除 Client 对象。

提示：有时不想删除 Client 对象，而只是想删除它所包含的属性并重新启动。可以通过将 expiration()方法设置为 0 秒（使用 client.expiration(0);语句）来删除存储在 Client 对象内的所有信息。

下一步，为 WebAdventureHome 程序的 Client 对象添加属性：

1．在文本编辑器或 HTML 编辑器中打开 HomePage.html 文件。

2．在</HEAD>标签的前面，输入下面的服务器段，它包含了一个简单的 if 语句，它使用逻辑非(!)运算符来检测 Client 对象内是否存在 sameSession 属性。若不存在 sameSession 属性，那么将创建它并将它的初始值设置为真。sameSession 属性用来指示客户正在同一个会话内运行，而不管它是否切换到应用程序内的另一个页面。在后面创建页面点击计数器时将使用此值。

```
<SERVER>
if (!client.sameSession) {
        client.sameSession = "true";
}
</SERVER>
```

3．保存 HomePage.html 文件。

Project 对象

Project 对象被用来保存应用程序的全局信息，它能够被所有使用应用程序的客户共享。每个应用程序具有自己的 Project 对象，在通过应用程序管理器中的 Start 按钮首次启动应用程序时将创建它。直到应用程序终止（使用应用程序中的 Stop 按钮）之前都能够访问应用程序的 Project 对象。

因为每个 Project 对象实例都试图保存应用程序的具体信息，所以 Project 对象未包含任何预定义的属性。使用语法 project.property = value;来为应用程序创建自己的属性。一个为 Project 对象创建的通用属性是某种用来标识客户的惟一编号。能够创建一个 Project 对象属性来跟踪上次被修改的编号，接着使用一个函数来更新编号并将它赋给客户。例如，可能想让 Web 应用程序接受在线订单。每次客户访问应用程序，都想赋给一个惟一的订单号。下面的代码检测 Project 对象的 lastInvoiceNum 属性的值并将它加 1。接着，在 Client 对象中创建一个新的 invoiceNum 属性并将 Project 对象的 lastInvoiceNum 属性的值赋给它。为了将订单号赋给访问页面的客户请在在线购物应用程序的首页中插入下面的代码。

```
<SERVER>
Project.lastInvoiceNum = ++Project.lastInvoiceNum;
Client.invoiceNum = Project.lastInvoiceNum;
</SERVER>
```

既然 Project 对象的属性能够被所有访问应用程序的客户访问，因此存在这种可能——在一个用户使用完属性之前，另一个用户可能试图使用它。若发生这种情况，那么可能遇到数据完整性问题。例如，若两个客户能够在同一时刻访问上面的代码，那么它们可能被赋给相同的订单号。为了在一个客户使用完 Project 对象的属性之前，防止另外一个客户访问它，可以使用 Project 对象的 lock()和 unlock()方法。lock()方法防止其他客户访问 Project 对象的属性，unlock()方法取消 lock()方法。在任何访问 Project 属性之前插入 project.lock();语句。在访问 Project 对象属性的最后一条语句后插入 project.unlock();语句。例如，为了防止 lastInvoiceNumber 代码实例出现数据完整性问题，如下使用 lock()和 unlock()方法：

```
<SERVER>
project.lock();
project.lastInvoiceNum = ++project.lastInvoiceNum;
client.invoiceNum = project.lastInvoiceNum;
project.unlock();
</SERVER>
```

因为对每个客户 Project 对象的属性都受保护，所以可以创建计算网址已被访问的次数的属性。计算网址已被访问次数通常被称为点击次数计数器。下一步，为 HomePage.html 文件添加点击次数计数器。

为 HomePage.html 文件添加点击次数计数器：

1．切换到文本编辑器或 HTML 编辑器中的 HomePage.html 文件。

2．在服务器段中，在 client.sameSession = "true";语句前添加下面的语句。第一条语句锁住 Project。if 语句使用逻辑非（!）运算符来检测 Project 对象的 counter 属性是否存在。counter 属性将保存此网页所接收到的点击的次数。若 counter 属性不存在，将创建它并将它初始化为 0。若 counter 属性存在，将它赋给 curNumber 变量，将它加 1。接着 curNumber

变量中的新值被赋给 counter 属性。最后，Project 被解锁。

```
project.lock();
if (!project.counter)
        project.counter = 1;
else {
        var curNumber = project.counter;
        curNumber = ++curNumber;
        project.counter = curNumber;
}
project.unlock();
```

3．下一步，在结束的</BODY>标签前添加下面的代码来显示此网页已被访问的次数。

```
<HR>
<B>This page has been accessed
<SERVER>write(project.counter);</SERVER> times.</B>
<HR>
```

4．保存 HomePage.html 文件，接着重新编译和启动 WebAdventureHome 程序。在 Navigator 中打开 StartPage.html 文件。在 First Name 文本框中输入姓名并单击“ Continue ”按钮打开 HomePage.html。此页的计数器应该从 1 开始。图 10-11 显示了此页被访问了数次之后的结果。

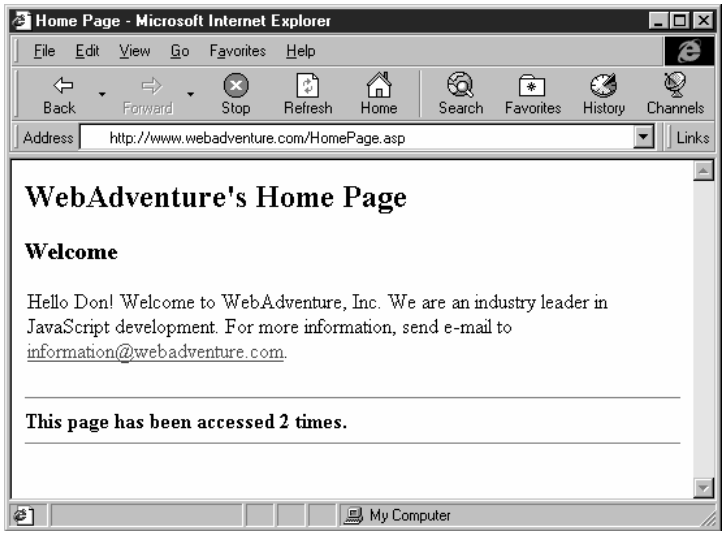


图 10-11　具有点击计数器的 HomePage.html

5．关闭 Web 浏览器窗口。

Server 对象

Server 对象包含了与正在运行的 Netscape 服务器相关的信息，并且在服务器首次启动时，它就开始存在。在服务器被停止之后，Server 对象将立即被销毁。若在同一台计算机上运行了多个服务器，那么接着每个服务器具有它自己的 Server 对象。Server 对象包含了几个能够被服务器上的所有应用程序访问的属性。表 10-3 列出了 Server 对象属性。

表 10-3　Server 对象属性

属　　性	描　　述
hostname	完整的主机名
host	名称、子域和域名
protocol	通信协议
port	所使用的端口号
jsVersion	版本和平台

与其他 LiveWire 对象一样，可以创建自定义的 Server 对象属性。使用 server.property = value;语法来创建 Server 对象属性。自定义 Server 对象属性的一个实例是 E-mail 地址，不同的应用程序使用它来提交状态报告或确认订单。

Server 对象包括了与 Project 对象所使用的相同的 lock()和 unlock()方法。若在改变 Server 对象值的服务器上具有多个应用程序，那么一定要使用 lock()和 unlock()方法来防止出现数据完整性问题。

10.1.5　创建客户簿

下面，将使用数个 LiveWire 对象为 WebAdventureHome 程序添加客户簿。将客户项作为 Project 对象的属性保存。注意在 Project 对象中保存客户项并不是必要的创建客户簿的最有效的方法，因为若需要重新启动服务器，那么清单将会丢失。更有效的方法是将客户项保存在数据库中。然而，此练习的主要目的是理解怎样使用核心的 LiveWire 对象。在第 11 章将学习怎样使用 LiveWire 来读取和写入数据库。

首先，往 HomePage.html 添加一些文字和一个表单：

1．在文本编辑器或 HTML 编辑器中打开 HomePage.html。

2．在结束的</BODY>前添加下面的信息标签和文字。文本中包括了 Project 对象的 guestCounter 属性，它返回客户簿上已签名的客户数。

```
<H3>Guest Book</H3>
Please join the other
<SERVER>write(project.guestCounter);</SERVER> people
who have signed our guest book. We have already
entered your first name for you. Please enter your
```

```
last name and e-mail address, then click the <B>I'm
Finished</B> button. If you would like to see a list
of other people who have signed our guest book, click
the <B>See Who's Been Here</B> button.
```

3．下一步，添加用于签名客户簿的表单。First Name 文本框使用 Request 对象将它的值设置为 StartPage.html 中此文本域的值。表单的 ACTION 属性打开 SignGuestBook.html 文件，它含有用来保存客户簿内的表单值的服务器端 JavaScript 代码。

```
<FORM METHOD="post" ACTION="SignGuestBook.html">
First Name: <INPUT TYPE="text" NAME="firstName" SIZE=10
VALUE='request.first'>
Last Name: <INPUT TYPE="text" NAME="lastName">
E-mail: <INPUT TYPE="text" NAME="email">
<INPUT TYPE="submit" VALUE=" I ’m Finished ">
</FORM>
```

4．创建另外一个表单，它将打开 ShowGuestBook.html 文件。ShowGuestBook.html 文件将包含用来显示已在客户簿上签名的客户的清单的服务器端 JavaScript 代码。

```
<FORM METHOD="post" ACTION="ShowGuestBook.html">
<INPUT TYPE="submit" VALUE=" See Who ’s Been Here ">
</FORM><P>
```

5．保存 HomePage.html。

下一步，创建 SignGuestBook.html 文件：

1．在文本编辑器或 HTML 编辑器中创建新文档。

2．输入文档的起始<HTML>和<HEAD>段。

```
<HTML>
<HEAD>
<TITLE>Guest Book</TITLE>
```

3．添加下面的服务器段。第一条语句锁定 Project，接着利用逻辑非（!）的 if 语句检测在 Project 对象是否已经创建 guestCounter 属性。若不存在，那么 if 语句将创建它。接着 guestCounter 属性的值被赋给 curGuestNum 变量，它被加 1。接着重新将它赋给 guestCounter 属性。将创建另外一个变量 curGuestInfo 来保存在客户簿上签名的客户的名、姓和 E-mail 地址。project["guest" + curGuestNum] = curGuestInfo;语句在 Project 对象中创建一个新属性用来保存当前客户的信息。注意通过将 guest 字符串和 curGuestNum 变量括在一对括弧中来实现创建属性名并将它赋给 Project 对象。服务器段中的最后一条语句对

Project 解锁。

```
<SERVER>
project.lock();
if (!project.guestCounter)
        project.guestCounter = 0;
var curGuestNum = project.guestCounter;
curGuestNum = ++curGuestNum;
project.guestCounter = curGuestNum;
var curGuestInfo = request.firstName + " "
        + request.lastName
        + ", " + request.email;
project ["guest" + curGuestNum ] = curGuestInfo;
project.unlock();
</SERVER>
```

4．添加下面的代码结束头部段，并开始主体段。

```
</HEAD>
<BODY>
```

5．输入下面的标题标签和表单，它使用 Request 对象的 firstName 属性为用户显示自定义的消息。表单包含了一个单独的按钮，它执行 history.back()方法返回到 HomePage.html 文件。

```
<H3>Thank you
<SERVER>write(request.firstName);</SERVER>! Your name
and e-mail address have been
added to our guest book.</H3>
<FORM>
<INPUT TYPE="button" VALUE="Return to WebAdventure's Home Page"
        onClick="history.back();">
<FORM>
```

6．添加下面的代码来结束<BODY>和<HTML>标签。

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

7．在 Data Disk 的 Tutorial.10 文件夹内将文件存为 SignGuestBook.html。

下一步，创建 ShowGuestBook.html 文件：

1．在文本编辑器或 HTML 编辑器内创建新文档。

2．输入文档的起始<HTML>、<HEAD>、<TITLE>和<BODY>段。

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

3．添加下面的信息文本和标签。

```
<H2>This is the Guest List</H2><HR>
<B>Name, E-Mail</B><BR>
```

4．输入下面的服务器段，它锁定和解锁 Project 并使用一个 for 循环来返回 Project 对象内的每个客户属性的内容。

```
<SERVER>
project.lock();
var numGuests = project.guestCounter;
for (var count = 1; count <= numGuests; ++count) {
        write(project ["guest" + count ] + "<BR>");
}
project.unlock();
</SERVER><HR>
```

5．输入下面的表单，它包含了一个单独的按钮来执行 history.back()方法返回到 HomePage.html 文件。

```
<FORM>
<INPUT TYPE="button" VALUE="Return to WebAdventure's
Home Page"
        onClick="history.back();">
</FORM>
```

6．添加下面的代码来结束<BODY>和<HTML>标签。

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

7．在 Data Disk 的 Tutorial.10 文件夹内将文件存为 SignGuestBook.html。

8．重新编译和启动 WebAdventureHome 应用程序。在使用 jsac 命令时确定包含了 4 个文件。

9．运行 StartPage.html 并输入名，接着单击“ Continue ”按钮打开 HomePage.html 文件。填写姓和 E-mail 地址，接着单击“ I’m Finished ”按钮来签署客户簿。图 10-12 显示了在 Navigator 中的 SignGuestBook.html。

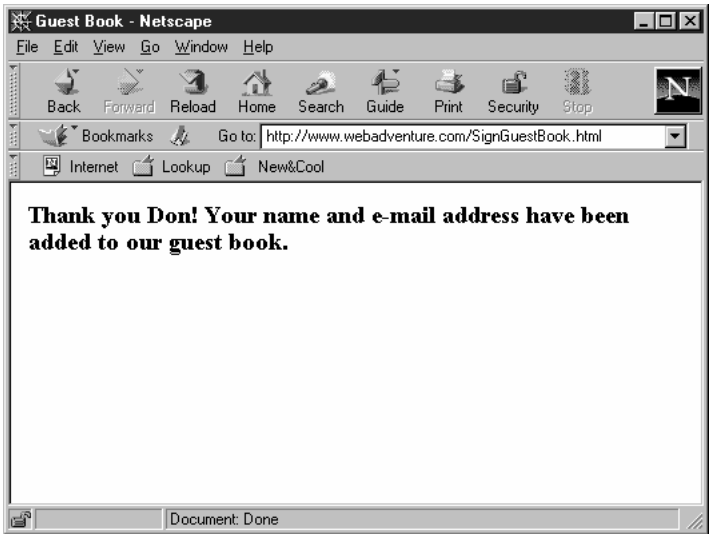


图 10-12　Navigator 中的 SignGuestBook.html

10．单击“ Return to WebAdventure’s Home Page ”按钮返回到 HomePage.html，接着单击“ See Who’s Been Here ”按钮来查看已在客户簿上签名的用户清单。图 10-13 显示了在数个客户签署了客户簿之后的 ShowGuestBook.html。

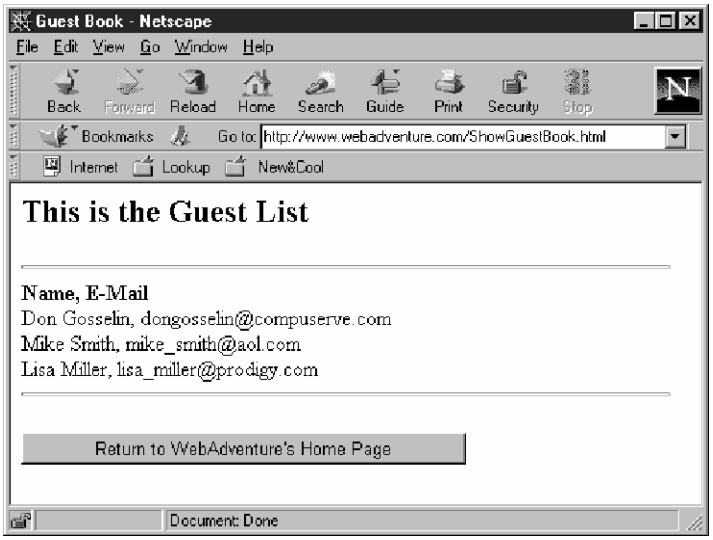


图 10-13　Navigator 中的 ShowGuestBook.html

11．关闭 Web 浏览器窗口。

10.1.6 总结

- ✧ Netscape 的服务器端 JavaScript 是众所周知的 LiveWire，Netscape Enterprise 和 FastTrack 服务器都支持它。
- ✧ 由客户机和服务器组成的系统是众所周知的两层系统。
- ✧ 三层、多层或客户/服务器系统由三个明显的部分组成：客户层、处理层和数据存储层。
- ✧ 服务器端 JavaScript 是基于客户端 JavaScript（能够处理客户层和数据存储层之间的通信）的编程语言。
- ✧ 使用服务器端 JavaScript 创建的 Web 应用程序并不是必须需要数据存储层。
- ✧ 不包括数据存储层的 Web 应用程序不是三层客户/服务器系统，它被称为分布式应用程序。
- ✧ 在使用 LiveWire 应用程序之前，必须对它进行编译和安装。
- ✧ 被编译成一个.web 文件的相关文件的集合称为 LiveWire 应用程序。
- ✧ <SERVER>…</SERVER>标签对被用来指明服务器端 JavaScript 代码。
- ✧ 若需要将服务器端 JavaScript 包括在 HTML 标签内，那么请使用反向引号（`）或 \Q 字符来封装它。
- ✧ 使用 jsac 程序来编译服务器端脚本。
- ✧ 应用程序管理器在 Web 服务器上安装和管理 LiveWire 应用程序。
- ✧ LiveWire 识别四种内嵌的对象：Request、Client、Project 和 Server 对象，它们负责处理层。在 LiveWire 中 Request、Client、Project 和 Server 对象被称为会话管理对象。
- ✧ Request 对象表示来自客户的当前 URL 请求。每次客户请求一个 URL，LiveWire 都将创建一个新的 Request 对象。
- ✧ Client 对象临时存储客户的具体信息，LiveWire 应用程序内的所有页面都能访问它。在客户首次访问给定应用程序内的 URL 时初始化 Client 对象。
- ✧ Project 对象被用来存储应用程序的全局信息，它能够被使用应用程序的所有客户共享。每个应用程序具有一个自己的 Project 对象，当利用应用程序管理器内的 Start 按钮首次启动应用程序时将创建它。
- ✧ lock()方法防止其他客户访问 Project 对象或 Server 对象的属性，unlock()方法取消 lock()方法。
- ✧ Server 对象包含了与正在运行的 Netscape 服务器相关的信息，在服务器首次启动时它就开始存在。

10.1.7　问题

1．由客户和服务器组成的系统被称为____。

a．主机结构

b．双系统结构

c．两层系统

d．广域网

2．通常什么是客户的主要作用？

a．定位符合请求的记录

b．高负荷的处理，如计算

c．数据存储

d．将接口展示给客户

3．在三层客户/服务器系统中，处理层不操作下面的哪种功能？

a．处理和计算

b．从数据存储层读入和写入信息

c．将任何信息返回给客户层

d．数据存储

4．下面哪种功能让客户去操作是安全的？

a．数据验证

b．数据存储

c．大量的处理

d．高负荷的计算

5．在发明服务器端 JavaScript 之前，开发三层客户/服务器应用程序的最常用的方法之一是什么？

a．CGI

b．Visual C++

c．Fortran

d．Visual Basic

6．使用____标签对将服务器端 JavaScript 添加到 LiveWire 应用程序的 HTML 文档中。

a．<LIVEWIRE>…</LIVEWIRE>

b．<SERVER>…</SERVER>

c．<SERVERSIDE>…</SERVERSIZE>

d．<LWSERVER>…</LWSERVER>

7．哪种是在 HTML 标签中包括 LiveWire 的 write()方法的正确语法？

a．\Qwrite("Hello World");\Q

b．\Qdocument.write("Hello World");\Q

c．/Qwrite("Hello World");/Q

d．/Qdocument.write("Hello World"); /Q

8．使用____命令行应用程序来编译 LiveWire 应用程序。

a．jcompile

b．jsac

c．lwcomp

d．lwc

9．下面哪种对象生存期最短？

a．Request

b．Client

c．Project

d．Server

10．下面哪项不创建 Request 对象？

a．客户端 JavaScript document.location 属性

b．客户端 JavaScript history.go()方法

c．服务器端 JavaScript redirect()方法

d．LiveWire 的应用程序管理器

11．怎样使用 Request 对象来引用“ password ”表单域？

a．request.password

b．request("password")

c．request.form("password")

d．client.request.password

12．哪种方法被用来改变 Client 对象的生存期？

a．lifespan()

b．expiration()

c．persistence()

d．duration()

13．在哪种对象中保存增加过的订单号，它必须能够被访问 LiveWire 应用程序的所有客户访问？

a．Request

b．Client

c．Project

d．Server

14．哪种方法防止其他客户访问 Project 对象或 Server 对象的属性？

a．session()

b．frozen()

c．lock()

d．preserve()

10.1.8　练习

1．Request 对象包括了 imageX 和 imageY 属性，它被用来创建服务器端的图像映射。在 Internet 中搜索所在的州、镇或城市的地图或图像。计算州地图上镇或城市周围 1 平方英寸区域的坐标。在用户单击州地图上的那块区域时，用所在的镇或城市的图像代替此区域。

2．创建一个通用的重定向页，用它向客户发送与所请求的不同的页面。可以将此页添加到 JavaScript 工具库内。使用 LiveWire 全局的 redirect()方法。

3．使用 Request 对象的属性来创建“ 客户信息 ”页，它显示客户浏览器的信息和 HTTP 信息。

4．使用 Server 对象的属性来创建 “ 服务器信息 ” 页，它显示服务器的信息。

5．参阅 Netscape 的 LiveWire 文档，查找与 File 对象相关的信息。File 对象被用来读写服务器上的文件。修改本节中创建的 WebAdventure 主页，将计数器和客户簿信息保存到服务器文件中。

6．阅读 Netscape 的 LiveWire 文档中的与应用程序管理器的调试功能相关的内容，接着撰写一篇怎样比较应用程序管理器调试功能和 Netscape 的 JavaScript 调试器应用程序的短论文。

7．LiveWire 可以使用 5 种方法来保存状态信息：Client-cookie、Client-url、Server-ip、Server-cookie 和 Server-url。在应用程序管理器的 Add Application 表单中的客户对象维护域中选择一种状态维护技术。阅读 Netscape 的 LiveWire 文档中有关状态维护的内容，接着撰写一篇有关每种状态维护技术的优点和缺点的短论文。

10.2　Microsoft Active Server Pages

本节目标

在本节将学习：

- ✧ Active Server Pages（ASP）
- ✧ 怎样创建 ASP 应用程序
- ✧ ASP 对象集合
- ✧ ASP 核心对象
- ✧ 怎样使用 ASP 创建客户簿

10.2.1 Active Server Pages 简介

Microsoft 的服务器端 JavaScript 是 Active Server Pages（ASP），它是 Microsoft Web 服务器内置的一个特性。像 LiveWire 一样，ASP 位于处理层并且根据客户的请求来执行。ASP 还能够与或不与数据源层一起使用。尽管在 LiveWire 和 ASP 之间存在区别，但是随着深入本节，将会注意到在这两种版本的服务器端 JavaScript 之间存在许多相似的地方，特别是在它们的核心对象之间。

在 ASP 与 LiveWire 之间的一个重要不同之处是 ASP 允许使用多种不同的脚本语言来创建服务器端程序。而 LiveWire 仅允许使用 JavaScript。默认情况下，ASP 支持 JavaScript（Microsoft 术语为 JScript）和 VBScript。还可以从第三方开发商那获得 Perl、REX 和 Python 脚本语言的 ASP 编译器。另外，可以在 ASP 脚本内混合使用不同语言，这样就可以利用某种语言所具有的惟一特性。因为重点是 JavaScript，所以将只讨论与 JavaScript 语言相关的特殊的 ASP 特性。

为了执行本节内的练习，需有能够使用 Windows 操作系统和 Microsoft Web 服务器的计算机。在 Windows NT Server 4.0 下运行的 Internet Information Server 3.0 及其高版本、在 Windows NT Workstation 4.0 下运行的 Peer Web Services 以及在 Windows 95/98 下运行的 Personnel Web Server 都支持 Active Server Pages。在本节内创建的实例和图使用的是运行 Peer Web Services 的 Windows NT Workstation 4.0。

ASP 不含有任何与 LiveWire 开发环境组件相似的东西。Active Service Pages 组件就是解释 ASP 代码的 JavaScript 和 VBScript 脚本引擎。不能像编译 LiveWire 应用程序那样编译 ASP 应用程序，也不需要安装、运行和停止 ASP 应用程序。ASP 自动识别 ASP 应用程序的修改并在下次客户请求它时重新编译应用程序。安装 ASP 应用程序只需将组成应用程序的文件拷贝到服务器的一个目录下。另外，在 Web 服务器首次接收到对应用程序的某个页面请求时自动启动应用程序。注意，术语“ LiveWire 应用程序 ”指编译过的后缀为.web 文件的集合，而“ ASP 应用程序 ”指在同一个根目录下相关 ASP 文件的集合。

提示：本教程仅提供 ASP 和服务器端 JavaScript 的简短的总体介绍。这里不讨论 Microsoft Web 服务器的网站管理功能。要了解有关 ASP 和 Microsoft Web 服务器更多的内容，请访问 Microsoft Developer Network：http://msdn.microsoft.com/。

10.2.2 创建 ASP 应用程序

与 LiveWire 应用程序不一样，使用后缀为.asp 的文件创建应用程序，在 LiveWire 应用程序中使用 HTML 文档或.js 源文件创建服务器端 JavaScript。ASP 文件是一个文本文件，与 HTML 文件相同，它能够包含客户端和服务器端的 JavaScript。在客户请求 ASP 文件时，

在将文档提交给客户之前 Web 服务器执行所有的服务器端 JavaScript。在客户的 Web 浏览器接收到文档之后，它立即执行客户端的 JavaScript。注意在客户向服务器请求一个 ASP 文件时，服务器都将编译此文件，而不管其中是否包含了服务器端 JavaScript 代码。因为编译过程需要额外的处理时间，而不需要对 HTML 文件进行编译，所以不要对不包含服务器端 JavaScript 代码的 HTML 文档使用.asp 后缀。

服务器端脚本

ASP 应用程序不像 LiveWire 那样使用<SERVER>…</SERVER>标签对来指明服务器端 JavaScript 代码。相反，ASP 使用脚本界定符<%和%>来指明服务器端 JavaScript 代码。定界符是用来标记代码段开始和结束的一个字符或一个字符序列。可以在界定符内包括任何对正在使用的脚本语言有效的命令。下面的代码中包含了一个脚本界定符实例，其中含有一条 Response 对象的 Write()方法。

```
<% Response.Write("Hello World"); %>
```

注意在前面的代码中 Write()方法添加在 Response 对象的后面，与在客户端 JavaScript 内将 write()方法添加在 Document 对象的后面一样。另外，请注意 Response 和 Write()方法首字符大写。在 ASP 中，具体的服务器端 JavaScript 对象和方法的首字符通常采用大写形式。在本节的后面部分将学习 Response 对象。除了 write()方法是将文本输出到浏览器，而 Write()方法将信息返回给客户之外，Write()方法的功能和客户端 write()方法的功能相同。

若需要在 HTML 标签内包括服务器端 JavaScript，那么可以使用脚本界定符。例如，在 ASP 中可以使用 Session 对象来创建自定义的客户属性，类似于 LiveWire 中的 Client 对象。假设已在 Session 对象中创建了一个 email 自定义属性，并想在使用 mailto:属性的链接内使用此属性。为了返回 ASP Session 对象内的 email 自定义属性并在链接中使用它，请使用与下面的代码类似的语法：

```
<A HREF="mailto:" + <% Response.Write(
Session.Contents("email")) %>>
Click here to send an e-mail</A>
```

注意在前面的代码中，将 Session 对象中的 email 属性作为 Contents 对象的参数引用的。ASP 中的属性存储在称为集合的数据结构中，而不是像 LiveWire 中那样作为对象的属性存储的。前面代码中引用的 Contents 对象实际上被称为 Contents 集合。在本节的最后将学习集合。

若客户端的某个 JavaScript 函数需要大量的处理，它对客户系统来说可能太繁重，那么可能需要指明<SCRIPT>…</SCRIPT>标签对让它在服务器上执行。RUNAT=SERVER 属性强制<SCRIPT>…</SCRIPT>标签对在服务器上运行。在包括了 RUNAT=SERVER 属性的<SCRIPT>…<SCRIPT>标签对内，能够包括任何服务器端 JavaScript 对象和方法。例如，图 10-14 使用 Request 对象的 Form 集的 language 属性向客户返回一条欢迎词。客户看不到

函数或代码——看到的只是 Response.Write()语言输出的结果。

```
<SCRIPT LANGUAGE=JScript RUNAT=SERVER>
function returnGreeting() {
        if (Request.Form("language") == "Spain")
                Response.Write("Buenos Dias");
        else if (Request.Form("language") == "Germany")
                Response.Write("Guten Tag");
        else if (Request.Form("language") == "Italy")
                Response.Write("Buon Giorno");
        else if (Request.Form("language") == "France")
                Response.Write("Bonjour");
        else
                Response.Write("I don't speak your language!");
}
</SCRIPT>
```

图 10-14 服务器端<SCRIPT>…</SCRIPT>标签对

下一步，利用 ASP 创建 WebAdventureHome 程序。首先创建首页文档。注意本节的步骤与在创建 LiveWire 应用程序所采取的步骤大部分相同。不需要重新输入这些步骤，可以拷贝和修改 LiveWire 程序中的文档。然而，一定要修改 LiveWire 命令以便与 ASP 命令语法一致，否则企图运行程序时将会接到错误。

为 WebAdventureHome 程序创建首页文档：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．输入文档的起始<HTML>、<HEAD>、<TITLE>和<BODY>段以及标题标签。

```
<HTML>
<HEAD>
<TITLE>Welcome</TITLE>
</HEAD>
<BODY>
<H2>Welcome to WebAdventure!</H2>
```

3．输入下面的 ASP 段，它创建一个 Date 对象，接着使用 toLocateString()方法将日期发送给客户。

```
<%
var curDate = new Date();
Response.Write("The current date and time are "
        + curDate.toLocaleString());
Response.Write("<HR>");
```

```
%>
```

4．添加下面的标题标签、包含了用于输入名的文本域和提交按钮（用来调用主 HTML 文档 HomePage.asp）的表单。使用提交按钮来将文本域内的值提交给 HomePage.asp。

```
<H3>Please enter your first name and click Continue to
proceed to our home page.<H3>
<FORM METHOD="post" ACTION="HomePage.asp">
First Name: <INPUT TYPE="text" NAME="first">
<INPUT TYPE="submit" VALUE=" Continue ">
</FORM>
```

5．添加下面的代码来结束<BODY>和<HTML>标签。

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

6．在 Data Disk 的 Tutorial.10 文件夹内将文件存为 StartPage.asp，接着将文件拷贝和上传到 ASP 服务器。

ASP 命令

ASP 命令用来指明在 ASP 文件中将内容输出到浏览器使用的脚本语言。有两种类型的 ASP 命令：处理命令和输出命令。ASP 处理命令用来向 Web 服务器提供怎样处理 ASP 文档内的脚本的信息。ASP 处理命令使用<%@…%>界定符创建。可以使用向 Web 服务器提供脚本相关信息的属性来创建处理命令。表 10-4 列出了处理命令属性。

表 10-4　处理命令属性

属　　性	描　　述
CODEPAGE	指明 ASP 文档的字符集
ENABLESESSIONSTATE	确定 ASP 文档是否维护状态信息
LANGUAGE	设置文档的默认脚本语言
LCID	设置脚本的本地化标识符
TRANSACTION	确定是否将脚本作为事务对待

默认情况下 ASP 使用 VBScript，所以为了在 ASP 应用程序中使用 JavaScript，必须包括处理命令。另外，处理命令必须被放置在 ASP 文件的第一行上，在<HTML>标签之前。例如，下面的代码创建了一个简单的 ASP 应用程序，它指明将 JScript（JavaScript）作为脚本语言并且禁止状态维护。

```
<%@ LANGUAGE=JScript ENABLESESSIONSTATE=false %>
```

```
<HTML>
HTML tags and script statements
</HTML>
```

输出命令将表达式的结果发送给 Web 浏览器（客户）。输出命令的语法是<% =表达式 %>。例如，若脚本包括了 favoriteColor 变量并且已将“ blue ”值赋给了它，接着输出命令<% =favoriteColor %>将文本“ blue ”发送给客户的 Web 浏览器。

提示：输出命令与 Response 对象的 Write()方法的功能相同。

图 10-15 显示了一个含有输出命令和处理命令的实例。实例将输出命令与 Response 对象的 Write()方法结合起来，并包括了处理命令用来指明将 JavaScript 作为默认的脚本语言。输出命令中的表达式计算 firstNum 和 secondNum，接着将文本“ The result of 10 minus 2 is 8 ”发送给用户 Web 浏览器，图 10-16 显示了输出结果。

```
<%@ LANGUAGE=JScript %>
<HTML>
<% var firstNum = 10, secondNum = 2;
Response.Write("The result of " + firstNum + " minus "
+ secondNum + " is "); %>
<%= firstNum  -  secondNum %>
</HTML>
```

图 10-15　ASP 命令实例

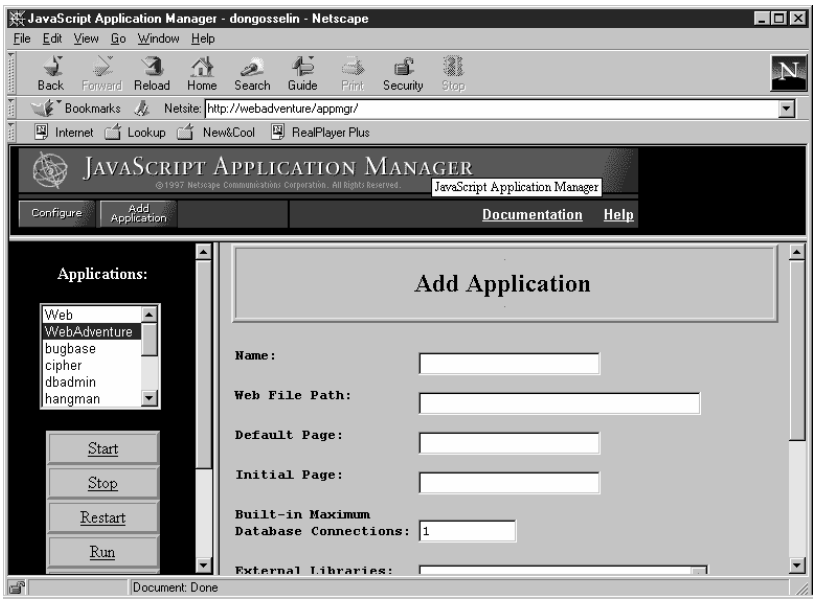


图 10-16　Web 浏览器中的 ASP 命令实例

下一步，将处理命令添加到 StartPage.asp 文档中：

1．返回到文本编辑器或 HTML 编辑器中的 StartPage.asp。

2．添加<%@ LANGUAGE=JScript %>

3．保存 StartPage.asp，接着将文件拷贝和上传到 ASP 服务器。

4．在 Internet Explorer 的 Address 框中输入 http://servername/StartPage.asp，用服务器名和保存 StartPage.asp 的服务器上的目录代替 servername。不能使用 Internet Explorer 的文件菜单下的打开命令将 StartPage.asp 作为本地文件打开。必须输入完整的 URL。

提示：可以使用随 Microsoft Web 服务器安装的 Personnel Web Server 程序配置 ASP 服务器的名称。

5．在打开文档之后，选中查看菜单下的源文件。在打开的文本编辑器窗口中，注意文档不包含服务器端 JavaScript 的 Response.Write()语句。而客户看到的仅是语句的输出结果。

6．关闭 Internet Explorer 和源文件窗口。

混合 HTML 和服务器端 JavaScript

在 Microsoft Web 服务器将包含了服务器端 JavaScript 的 ASP 文档传递给客户之前，它先执行所有脚本界定符的内容。若用户在接收到 ASP 文档之后查看源文档，将看不到任何它们所包含的<%…%>脚本界定符或 JavaScript 代码。客户机仅显示代码返回的结果。例如，图 10-17 显示了一个含有客户端和封装在脚本界定符内的服务器端 JavaScript 的 ASP 文档。此例假设 Session 对象的 Contents 集合内的 email 属性值是 donggosselin@compuserve.com。

在图 10-17 中可以看到脚本界定符可以分散在 HTML 标签和文本中。然而，甚至可以将 HTML 标签和文本作为服务器端 JavaScript 选择结构的一部分，如 if…else 语句。例如在下面的代码中，因为每个 if 或 else if 语句封装在脚本界定符内，所以不需要使用 Response 对象的 Write()方法来输出正确的欢迎词。基于 if…else 结构的结果仅返回一条欢迎词。

```
<% if (Request.Form("language") == "Spain") %>
        Buenos Dias
<% else if (Request.Form("language") == "Germany") %>
        Guten Tag
<% else if (Request.Form("language") == "Italy") %>
       Buon Giorno
<% else if (Request.Form("language") == "France") %>
        Bonjour
<% else %>
        I don't speak your language!
```

还可以在一个脚本界定符内编写代码，如下面的代码所示。然而，欢迎词并不作为

HTML 标签或文本对待，因为它们封装在脚本界定符内。因此，每条欢迎词必须使用 Response.Write()语句来将自己返回给客户。

```
<%@ LANGUAGE=JScript %>
<HTML>
<HEAD>
<TITLE>Client and Server-Side JavaScript</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
document.write("<H2>This line is generated by client-side
JavaScript.</H2>");
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
<BODY>
<% Response.Write("<H2>This line is generated by server-side
JavaScript.</H2>"); %>
<A HREF="mailto:" + <% Session.Contents("email") %>>
Click here to send an e-mail to Don Gosselin</A>
</BODY>
</HTML>
```

图 10-17　服务器上具有客户和服务器端 JavaScript 的 ASP 文档

```
<%
if (Request.Form("language") == "Spain")
        Response.Write("Buenos Dias");
else if (Request.Form("language") == "Germany")
        Response.Write("Guten Tag");
else if (Request.Form("language") == "Italy")
        Response.Write("Buon Giorno");
else if (Request.Form("language") == "France")
        Response.Write("Bonjour");
else
        Response.Write("I don't speak your language!");
%>
```

HTML 脚本代码如图 10-18 所示。

图 10-19 显示了另外一个自定义 Web 页的实例，它在服务器端 JavaScript 内分散插入了 HTML 标签和文本。此例显示了一个根据客户请求的信息来创建 HTML 文档。假设发出客户请求的文档含有一个 clientName 表单域。此例使用了 Request 对象的 From 集的 clientName 属性，并且还在返回给客户的整个文本内分散插入了输出命令，用它来创建自

定义的响应。此例还假设企业的 E-mail 地址保存在 Application 对象的 Contents 集的 email 属性中。图 10-20 显示了 Web 浏览器中输出结果，假设 Don 是 clientName 属性的值。

```
<HTML>
<HEAD>
<TITLE>Client and Server-Side JavaScript</TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
document.write("<H2>This line is generated by client-side
JavaScript.</H2>");
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
</HEAD>
<BODY>
<H2>This line is generated by server-side JavaScript.</H2>
<A HREF="mailto:"dongosselin@compuserve.com" + >
Click here to send an e-mail to Don Gosselin</A>
</BODY>
</HTML>
```

图 10-18　客户端上的具有客户端和服务器端 JavaScript 的 ASP 文档

```
<%@ LANGUAGE=JScript %>
<HTML>
<HEAD>
<TITLE>HTML and Server-Side JavaScript</TITLE>
</HEAD>
<BODY>
Dear <%= Request.Form("clientName") %>:<P>
Thank you for your interest in our Web development services.
As you know, it is difficult to keep up with rapidly changing
technology. With WebAdventure's help, you can be confident
that your Web site will be on the cutting edge. Remember,
<%= Request.Form("clientName") %>, WebAdventure is here for
you. For more information, click <A HREF="mailto:" +
<%= Application.Contents("email") %>>here</A> to send us an
e-mail.
</BODY>
</HTML>
```

图 10-19　用 HTML 点缀的服务器端 JavaScript

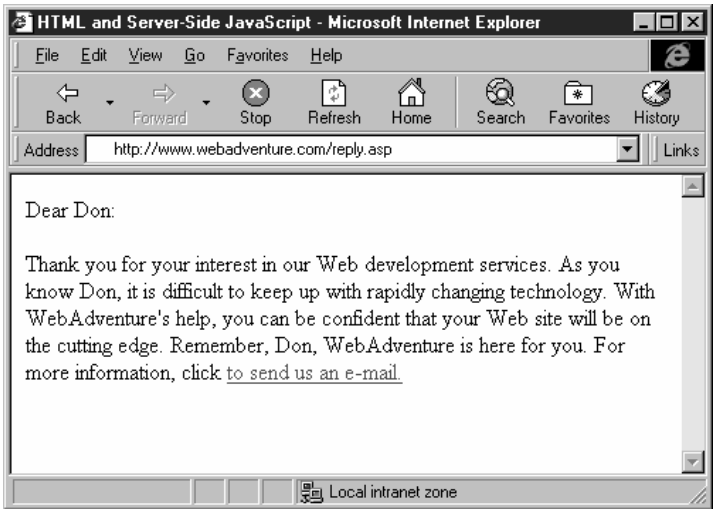


图 10-20　Web 浏览器中用 HTML 点缀的服务器端 JavaScript

10.2.3　对象集合

在学习 ASP 核心对象之前，需要理解集合。集合类似于在 ASP 核心对象中存储变量的数组。可以将集合想象为类似于 LiveWire 核心对象中的属性。将变量赋给集合的语法是 object.collection("variable") = value;。使用 object.collection("variable");语法来引用 ASP 代码中的变量。例如，在 Application 和 Session 对象中都分别包含了 Contents 集合，它用来保存自定义的变量。下面的代码在 Session 对象的 Contents 集合中创建了一个 name 变量并给它赋 Don Gosselin 值，接着使用 Response.Write()语句将值赋送给客户浏览器。

```
<%
Session.Contents("name") = "Don Gosselin";
Response.Write("Your name is " + Session.Contents("name"));
%>
```

提示 在 Visual Basic 编程中常用集合。ASP 的非 JavaScript 语法的大部分来源于 Visual Basic 编程，因为 ASP 的默认脚本语言是 VBScript，所以它从 Visual Basic 继承而来。

若赋给集合的变量 name 在整个对象集合内是惟一的，那么在代码中引用对象时可以省略集合名。例如，若 Session 对象在所有它的集合中仅包含了一个 name 变量，那么可以使用类似于 Response.Write(Session("name"));语句来在代码中引用 name 变量。然而，为了消除关于哪个集合包含了指定的变量的不确定性，包括集合名通常是安全的。另外，若后来在同一个对象中为另外一个集合添加相同的变量名，那么省略集合名可能导致错误。因此，在本教程代码中引用变量时都列出了集合名。

集合中的每个变量被依次编号，类似于给数组赋下标。与数组下标不同，集合编号从 1 开始，而数组下标从 0 开始。可以使用编号而不是变量名来引用集合变量。例如，下面的代码在 Session 对象的 Contents 集合中创建了三个变量：firstName、lastName 和 email，接着使用每个变量的集合编号来将值返回给客户。

```
<%
Session.Contents("firstName") = "Don";
Session.Contents("lastName") = "Gosselin";
Session.Contents("email") = "dongosselin@compuserve.com";
Response.Write(Session.Contents(1));
Response.Write(Session.Contents(2));
Response.Write(Session.Contents(3));
%>
```

注意若从集合中删除项时，赋给集合中变量的编号可能改变。不能假设一个编号始终代表集合中同一个变量。

提示：可以使用 Contents 集合的 Remove()和 RemoveAll()方法从集合中删除项。Application 对象和 Session 对象都包含了 Contents 集合。

ASP 集合支持 Count 属性，它返回集合中变量的个数。可以在循环语句中使用 Count 属性来遍历集合中的变量。例如，下面的代码使用 for 循环将 Contents 集合中的变量返回给客户。for 循环中的条件表达式比较 curVariable 变量和 Count 属性。当 curVariable 小于或等于 Count 属性时，for 循环继续遍历集合。

```
<%
Session.Contents("firstName") = "Don";
Session.Contents("lastName") = "Gosselin";
Session.Contents("email") = "dongosselin@compuserve.com";
for (var curVariable = 1;
curVariable <= Session.Contents.Count;
        ++curVariable) {
        Response.Write(Session.Contents(curVariable));
}
%>
```

10.2.4　ASP 核心对象

像 LiveWire 一样，ASP 包括了数个在处理层运行的内置对象。ASP 核心对象是 Request、Response、Session、Application 和 Server 对象。使用每种对象来访问处理层中的特殊类型

的信息并且在服务器上保存状态信息。

提示：注意本节中的许多实例与在 10.1 节中看到的实例相同。我们使用相同的实例来演示怎样使用两种不同版本的服务器端 JavaScript 来完成同一样任务。

Request 对象

ASP 的 Request 对象代表了来自客户的当前 URL 请求，它与 LiveWire 的 Request 对象等价。每次客户请求一个 URL，ASP 都将创建一个新的 Request 对象。例如，若用户单击浏览器中的链接或新的 URL，那么 ASP 将创建一个 Request 对象。在客户端的 JavaScript 使用 document.location 或 history.go()方法时 ASP 也创建一个 Request 对象。在所有 ASP 内置对象中 Request 对象的生存期最短，因为它仅在当前请求完成之前才存在。

Request 对象包含了数个用来保存客户请求信息的集合。表 10-5 列出了 Request 对象的集合。

表 10-5　Request 对象集合

集　　合	内　　容
ClientCertificate	随请求发送的客户验证的字段值
Cookies	随请求发送的 Cookies
Form	浏览器所显示的文档中的命名表单元素的值
QueryString	在查询串中添加给 URL 的“名称=值”对
ServerVariables	环境变量

提示：ASP 的 Request 对象还包括了 TotalBytes 属性（它返回在客户请求中所发送的总字节数）和 BinaryRead 方法（它检索作为 POST 请求的一部分、从客户发送给服务器的数据）。

ASP 的 Request 对象的 Form 集合包含了代表来自正在请求的 Web 页的表单元素变量。ASP 接收用户浏览器上表单内的所有命名元素并将它们作为变量添加到 Request 对象的 Form 集合。回顾一下在单击表单的提交按钮时，表单上的每个域都被以“名称=值”对的形式提交给服务器。“名称=值”对的名称部分成为 Form 集合中的变量名，并且值部分被作为变量值赋给变量。图 10-21 显示了一个典型的表单。

在将图 10-21 中的表单提交给 ProcessOrder.asp ASP 文档之前，域名和值作为变量赋给 Request 对象的 Form 集合。使用下面的语句来引用 Request 对象的 Form 集合中的每个表单变量：

```
Request.Form("name")
Request.Form("address")
Request.Form("city")
Request.Form("state")
Request.Form("zip")
```

Request.Form("email")

```
<HTML>
<HEAD>
<TITLE>Customer Information</TITLE>
</HEAD>
<BODY>
<H2>Customer Information</H2>
<FORM METHOD="post" ACTION="ProcessOrder.asp"
NAME="customer_information">
Name<BR>
<INPUT TYPE="text" NAME="name" SIZE=50><BR>
Address<BR>
<INPUT TYPE="text" NAME="address" SIZE=50><BR>
City, State, Zip<BR>
<INPUT TYPE="text" NAME="city" SIZE=38>
<INPUT TYPE="text" NAME="state" SIZE=2 MAXLENGTH=2>
<INPUT TYPE="text" NAME="zip" SIZE=5 MAXLENGTH=5><BR>
E-Mail<BR>
<INPUT TYPE="text" NAME="email" SIZE=50><P>
<INPUT TYPE="reset">
<INPUT TYPE="submit">
</BODY>
</HTML>
```

图 10-21　一个典型的表单

在“名称=值”对被作为查询字符串添加给 URL 时，它们作为变量赋给 Request 对象的 QueryString 集合中。思考一下下面的代码，它将查询字符串添加在 URL 的后面。

```
<A HREF="http://www.URL.com/TargetPage.asp?firstName=Don
&lastName=Gosselin&occupation=writer">Link Text</A>
```

在用户单击此链接之后，将打开 TargetPage.asp。此后任何 ASP 代码都能够将 firstName、lastName 和 occupation 作为 Request 对象的 QueryString 集合中的变量来引用，如下：

```
Request.QueryString("firstName")
Request.QueryString("lastName")
Request.QueryString("occupation")
```

既然理解了 Request 对象的基本知识，那么将主 WebAdventure 文档作为 ASP 文档创建。

将主 WebAdventure 文档作为 ASP 文档创建：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．输入文档的起始处理命令和<HTML>、<HEAD>、<TITLE>和<BODY>段。

```
<%@ LANGUAGE=JScript %>
<HTML>
<HEAD>
<TITLE>Home Page</TITLE>
</HEAD>
<BODY>
```

3．添加下面的标题标签。

```
<H2>WebAdventure ’s Home Page</H2>
<H3>Welcome</H3>
```

4．输入下面的段落，它使用 Request 对象和输出命令来插入 StartPage.asp 文件中的名文本域的值。

```
Hello <%= Request.Form("first") %>! Welcome to
WebAdventure, Inc. We are an industry leader in
JavaScript development. For more information, send
e-mail to <A HREF="mailto:information@webadventure.com">
information@webadventure.com</A>.<P>
<HR>
```

5．添加下面的代码来结束<BODY>和<HTML>标签。

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

6．将文件存为 HomePage.asp，接着将文件拷贝和上传到 ASP 服务器。

7．在 Web 浏览器中打开来自 ASP 服务器的 StartPage.asp。在 First Name 文本框中输入名并单击“ Continue ”按钮来打开 HomePage.asp。图 10-22 显示了文件在 Internet Explorer 中显示的结果。可以看到在 StartPage.asp 中输入的名已被插入到 HomePage.asp 中。

8．关闭 Web 浏览器窗口。

Response 对象

Response 对象将输出和信息返回给客户。已经使用过 Response 对象的 Write()方法将文本返回给客户。尽管本教程仅讨论 Write()方法，但是 Response 对象还包括了数个其他方法，它们与属性一样在构造返回给客户的响应时十分有用。例如，Response 对象的 Redirect()

方法将客户引导至另外一个不同的 Web 页，它与客户端 JavaScript 的 Location 对象的 href 属性功能一样。

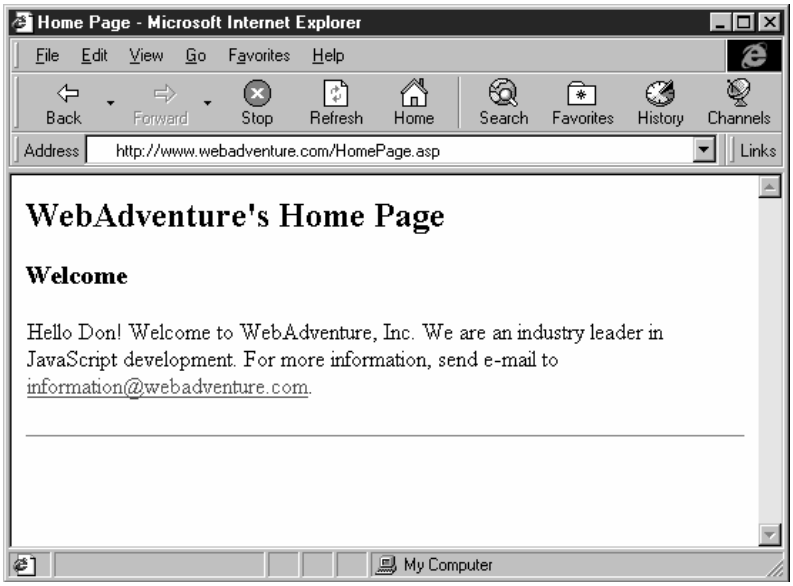


图 10-22　Internet Explorer 中的 HomePage.asp

Response 对象是 ASP 仅有的。LiveWire 仅使用全局的 write()方法来将输出返回给客户。注意 Response 对象的许多方法和属性与 LiveWire 的全局方法相同。

提示：请访问 Microsoft Developer Network 的网址 http://msdn.microsoft.com/来阅读完整的 Response 对象的属性和方法清单。

提示：Response 对象还包括了一个 Cookies 集合，它被用来设置客户系统上的 Cookies。Cookies 集合是包含在 Response 对象中的惟一一种集合。

Session 对象

每次客户请求 ASP URL 时都将初始化一个新的 Request 对象，接着在将 URL 传递给客户之后就立即销毁它。与 LiveWire 一样，一个 ASP 应用程序可能由多个文档组成。因为在 URL 被传递给客户之后就立即销毁 Request 对象，所以不能够与应用程序中的其他不同的页面使用同一个 Request 对象。若想跨 ASP 应用程序中的多个页面保存客户信息，那么必须使用 Session 对象。Session 对象被用来临时保存指定的客户信息并且 ASP 应用程序中的所有页面都能够访问它。ASP 的 Session 对象与 LiveWire 的 Client 对象等价。在一个客户首次访问给定应用程序中的某个 URL 时都将初始化一个 Session 对象。

使用 Session 对象的 Contents 集合来保存应用程序中客户的相关信息与将用户信息作为 LiveWire 的 Client 对象的属性保存的方式一样。例如，可以创建一个包含了用户名的变量并且使用 name 变量为用户所访问的应用程序的每个页面创建一个自定义的欢迎词。Session 对象变量的其他实例包括用于在线商店的购物车清单以及保存由多部分组成的表单内的域

值。假设下面的代码是由多部分组成的表单内的首页。代码显示了 ASP 应用程序的一个实例，它将表单的域值作为 Session 对象的 Contents 集合的变量进行赋值。在生成的由多部分组成的表单的第二个页面的 HTML 文档中插入此代码。在用户请求第二个页面时执行此代码。在此例中，从 Request 对象的 Form 集合中的相关变量中检索每个表单元素值。

```
<%
Session.Contents("name") = Request.Form("name");
Session.Contents("address") = Request.Form("address");
Session.Contents("city") = Request.Form("city");
Session.Contents("state") = Request.Form("state");
Session.Contents("zip") = Request.Form("zip");
Session.Contents("email") = Request.Form("email");
%>
```

提示：Session 对象还包括了 StaticObjects 集合，它代表 HTML 文档的<OBJECT>标签所创建的对象。

Session 对象还包括了数个在表 10-6 中列出的内置属性。

表 10-6 Session 对象属性

属　性	描　述
CodePage	指定语言所使用的字符集
LCID	标识用户所在地区的或应用程序的首选语言
SessionID	用户的会话标识
Timeout	Session 对象的生存期

Session 对象的一个特别有用的属性是 Timeout 属性，它决定了 Session 对象的生存期。Timeout 属性与 LiveWire 的 expiration()方法等价。与 LiveWire 的 Client 对象一样，Session 对象的默认生存期为 10 分钟。Timeout 属性的语法是 Session.Timeout = 分钟;。注意时间是以分钟指定的，而不像 LiveWire 的 expiration()方法是以秒指定的。若想将 Session 对象的生存期增加为 20 分钟，那么请使用 Session.Timeout = 20;语句。

提示：Session 对象的 Contents 集合还包括三种方法：Abandon()、Remove()和 RemoveAll()。Abandon()方法完全销毁一个 Session 对象。Remove()和 RemoveAll()方法被用来从 Session 对象集合中删除项。

下一步，将为 WebAdventureHome 程序的 Session 对象添加一个属性：

1．在文本编辑器或 HTML 编辑器中打开 HomePage.asp。

2．在</HEAD>标签的前面，输入下面的服务器段，它包含了一个简单的 if 语句，它使用逻辑非（!）运算符来检测 Session 对象内是否存在 sameSession 属性。若不存在 sameSession 属性，那么将创建它并将它的初始值设置为真。sameSession 属性用来指示客

户正在同一个会话内运行，而不管它是否切换到应用程序内的另一个页面。在后面创建页面点击计数器时将使用此值。

```
<%
if (!Session.Contents("sameSession")) {
        Session.Contents("sameSession") = "true";
}
%>
```

3．保存 HomePage.asp 文件，接着将它拷贝和上传到 ASP 服务器。

Application 对象

Application 对象被用来保存应用程序的全局信息，它能够被使用应用程序的所有客户共享。ASP 的 Application 对象与 LiveWire 的 Project 对象等价。每个应用程序具有自己的 Application 对象。在客户首次访问应用程序的一个页面时，将自动启动一个 ASP 的 Application。ASP 的 Application 一直运行直到服务器被关闭。

在 Application 的 Contents 集合中为它创建自己的变量。为 Application 对象创建的一类通用属性是某类用来标识客户的惟一编号。可以在 Application 对象的 Contents 集合中创建一个用来保存上次编号的变量，接着使用一个函数来更新编号并且将它赋给客户。例如，存在一个能够接受在线订单的应用程序。每个客户访问应用程序时，都想赋给他一个惟一的订单编号。下面的代码将 Application 对象的 Contents 集合中的 lastInvoiceNum 变量的值赋给 curInvoiceNum 变量。接着将 curInvoiceNum 加 1。在 Session 对象的 Contents 集合中创建一个新的 invoiceNum 变量并将 curInvoiceNum 变量的值赋给它。可以将此代码插入在线订购应用程序的首页中，用它来将订单号赋给访问此页面的客户。

```
<%
curInvoiceNum = Application.Contents("lastInvoiceNum");
curInvoiceNum = ++curInvoiceNum;
Application.Contents("lastInvoiceNum") = curInvoiceNum;
Session.Contents("invoiceNum") = curInvoiceNum;
%>
```

提示：Application 对象还能够访问 Session 对象的 StaticObject 集合，它代表 HTML 文档的<OBJECT>标签所创建的对象。

提示：可以使用 Contents 集合的 Remove()和 RemoveAll()方法从 Application 对象集合中删除项。

ASP 的 Application 对象的变量能够被所有访问应用程序的客户访问，因此存在这种可能——在一个客户使用完它之前，另一个用户可能试图访问它。为了在一个客户使用完 Application 对象的变量之前，防止另外一个客户访问它，可以使用 Application 对象的 Lock()和 Unlock()方法，它们执行与 LiveWire 的 Lock()和 Unlock()方法相同的功能。Lock()方法

防止其他客户访问 Application 对象的属性，Unlock()方法取消 Lock()方法。在任何访问 Application 对象属性之前插入 Application.Lock();语句。在访问 Application 对象属性的最后一条语句后插入 Application.Unlock();语句。例如，为了防止 lastInvoiceNumber 代码实例出现数据完整性问题，如下使用 Lock()和 Unlock()方法：

```
<%
Application.Lock();
curInvoiceNum = Application.Contents("lastInvoiceNum");
curInvoiceNum = ++curInvoiceNum;
Application.Contents("lastInvoiceNum") = curInvoiceNum;
Session.Contents("invoiceNum") = curInvoiceNum;
Application.Unlock();
%>
```

既然理解了怎样保护跨多个用户会话的变量，那么将为 HomePage.asp 文件添加点击次数计数器。

为 HomePage.asp 文件添加点击次数计数器：

1．切换到文本编辑器或 HTML 编辑器中的 HomePage.asp。

2．在 ASP 段中，在 Session.Contents("sameSession") = "true";语句前添加下面的语句。第一条语句锁住 Application。if 语句使用逻辑非(!)运算符来检测 Application 对象的 counter 变量是否存在。counter 变量将保存此网页所接收到的点击的次数。若 counter 变量不存在，将创建它并将它初始化为 1。若 counter 变量存在，将它赋给 curNumber 变量，并将它加 1。接着 curNumber 变量中的新值被赋给 counter 变量。最后，Application 被解锁。

```
Application.Lock();
if (!Application.Contents("counter"))
        Application.Contents("counter") = 1;
else {
        var curNumber = Application.Contents("counter");
        curNumber = ++curNumber;
        Application.Contents("counter") = curNumber;
}
Application.Unlock();
```

3．下一步，在结束的</BODY>标签前添加下面的代码来显示此网页已被访问的次数。

```
<B>This page has been accessed <%=
Application.Contents("counter")
%> times.</B>
<HR>
```

4．保存 HomePage.asp，接着将它拷贝和上传至 ASP 服务器。

5．在 Internet Explorer 中打开来自 ASP 服务器的 StartPage.asp。在 First Name 文本框中输入名并单击“ Continue ”按钮打开 HomePage.asp。此页的计数器应该从 1 开始。图 10-23 显示了此页被访问了两次之后的结果。

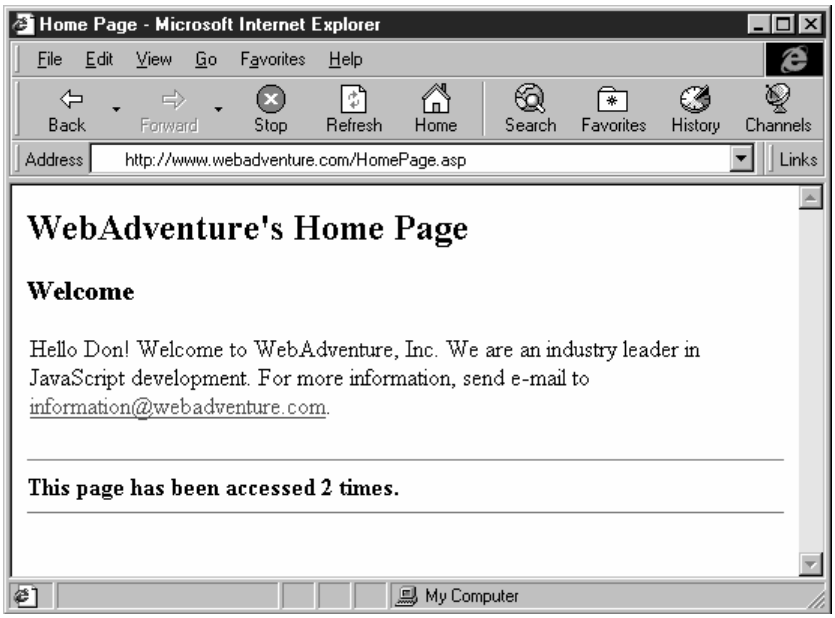


图 10-23　具有点击计数器的 HomePage.asp

6．关闭 Web 浏览器窗口。

Server 对象

Server 对象让 ASP 的应用程序能够访问能够被服务器上的所有应用程序访问的属性和方法。ASP 的 Server 对象不包含集合，并且不能为它添加自定义变量。与 LiveWire 的 Server 对象相比，ASP 的 Server 对象主要用于管理 ASP 应用程序在服务器上的行为。

ASP 的 Server 对象包含了的惟一一个属性——ScriptTimeout。ScriptTimeout 决定了在服务器停止 ASP 应用程序之前它能够运行的时间。默认情况下，服务器允许 ASP 应用程序运行的时间为 90 秒。若想让应用程序运行更长的时间，那么请使用 Server.ScriptTimeout = 秒;语法来增加它的运行时间。例如，为了将应用程序的最大运行时间增至 2 分钟，请使用 Server.ScriptTimeout = 120;语句。Server 对象还包括了在表 10-7 中列出的方法。

表 10-7　Server 对象方法

方　法	描　述
CreateObject()	初始化服务器组件
Execute()	执行 ASP 文件
GetLastError()	返回 ASPError 对象，它描述了所发生的错误状态

续表

方　法	描　述
HTMLEncode()	使用 HTML 编码对指定的字符串进行编码
MapPath()	将相对路径或虚拟路径映射为相应的服务器物理目录
Transfer()	将当前 ASP 的集合变量和对象传递给其他 ASP 应用程序
URLEncode()	使用 URL 编码格式化字符串

提示：在第 11 章中将使用服务器对象的几个方法来访问数据库。

10.2.5 创建客户簿

下一步，将为 ASP 版的 WebAdventureHome 程序添加客户簿，把客户项作为 Application 对象的 Contents 集合中的变量保存。与 LiveWire 一样，在核心对象中保存客户项并不是创建客户簿的最有效的方法，因为若需要重新启动服务器，那么清单将会丢失。更有效的方法是将客户项保存在数据库中。然而，此练习的主要目的是理解怎样使用 ASP 核心对象。在第 11 章将学习怎样使用 ASP 来读取和写入数据库。

首先，往 HomePage.asp 添加一些文字和一个表单：

1．在文本编辑器或 HTML 编辑器中打开 HomePage.asp。

2．在结束的</BODY>前添加下面的信息标签和文字。文本中包括了 Application 对象的 guestCounter 变量，它返回客户簿上已签名的客户数。

```
<H3>Guest Book</H3>
Please join the other <%=
Application.Contents("guestCounter") %> people who have
signed our guest book. We have already entered your
first name for you. Please enter your last name and
e-mail address, then click the <B>I'm Finished</B> but-
ton. If you would like to see a list of other people
who have signed our guest book, click the <B>See Who's B
een Here</B> button.
```

3．下一步，添加用于用户签署客户簿的表单。First Name 文本框使用 Request 对象将它的值设置为 StartPage.asp 中此文本域的值。表单的 ACTION 属性打开 SignGuestBook.asp 文件，它将包含用来保存客户簿内的表单值的服务器端 JavaScript 代码。

```
<FORM METHOD="post" ACTION="SignGuestBook.asp">
First Name: <INPUT TYPE="text" NAME="firstName" SIZE=10
VALUE=<%= Request.Form("first") %>>
```

```
Last Name: <INPUT TYPE="text" NAME="lastName">
E-mail: <INPUT TYPE="text" NAME="email">
<INPUT TYPE="submit" VALUE=" I 'm finished ">
</FORM>
```

4．创建另外一个表单，它将打开名为 ShowGuestBook.asp 的文件。ShowGuestBook.asp 文件将包含用来显示已在客户簿上签名的客户的清单的服务器端 JavaScript 代码。

```
<FORM METHOD="post" ACTION="ShowGuestBook.asp">
<INPUT TYPE="submit" VALUE=" See Who 's Been Here ">
</FORM><P>
```

5．保存 HomePage.asp，并将文件拷贝和上传到 ASP 服务器上。

下一步，创建 SignGuestBook.asp 文件：

1．在文本编辑器或 HTML 编辑器中创建新文档。

2．输入文档的处理命令和起始<HTML>和<HEAD>段。

```
<%@ LANGUAGE=JScript %>
<HTML>
<HEAD>
<TITLE>Guest Book</TITLE>
```

3．添加下面的 ASP 段。第一条语句锁定 Application，接着利用逻辑非（!）的 if 语句检测在 Application 对象中是否已经创建 guestCounter 变量。若它不存在，那么 if 语句将创建它。接着 guestCounter 变量的值被赋给 curGuestNum 变量，将它加 1，接着重新将它赋给 guestCounter 变量。将创建另外一个变量 curGuestInfo 来保存在客户簿上签名的客户的姓、名和 E-mail 地址。Application.Contents("guest"+ curGuestNum) = curGuestInfo;语句在 Application 对象的 Contents 集合中创建一个新属性用来保存当前客户的信息。服务器段中的最后一条语句对 Application 解锁。

```
<%
Application.Lock();
if (!Application.Contents("guestCounter"))
        Application.Contents("guestCounter") = 0;
var curGuestNum = Application.Contents("guestCounter");
curGuestNum = ++curGuestNum;
Application.Contents("guestCounter") = curGuestNum;
var curGuestInfo = Request.Form("firstName") + " "
        + Request.Form("lastName")
```

```
        + ", " + Request.Form("email");
Application.Contents("guest" + curGuestNum) =
curGuestInfo;
Application.Unlock();
%>
```

4．添加下面的代码结束头部段，并开始体段。

```
</HEAD>
<BODY>
```

5．输入下面的标题标签和表单，它使用 Request 对象 Form 集合中的 firstName 属性为用户显示自定义的消息。表单含有惟一一个按钮，它执行 history.back()方法来返回到 HomePage.asp。

```
<H3>Thank you <%= Request.Form("firstName") %>! Your
name and e-mail address have been added to our guest
book.</H3>
<FORM>
<INPUT TYPE="button" VALUE="Return to WebAdventure's
Home Page"
        onClick="history.back();">
</FORM>
```

6．添加下面的代码来结束<BODY>和<HTML>标签。

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

7．在 Data Disk 的 Tutorial.10 文件夹内将文件存为 SignGuestBook.asp，并将它拷贝和上传到 ASP 服务器上。

最后，创建 ShowGuestBook.asp 文件：

1．在文本编辑器或 HTML 编辑器内创建新文档。

2．输入处理命令和文档的起始<HTML>、<HEAD>、<TITLE>和<BODY>段。

```
<%@ LANGUAGE=JScript %>
<HTML>
<HEAD>
<TITLE>Guest Book</TITLE>
```

```
</HEAD>
<BODY>
```

3．添加下面的信息文本和标签。

```
<H2>This is the Guest List</H2><HR>
<B>Name, E-Mail</B><BR>
```

4．输入下面的服务器段，它锁定和解锁 Application 并使用一个 for 循环来返回 Application 对象的 Contents 集合内的每个客户变量的内容。

```
<% Application.Lock();
var numGuests = Application.Contents("guestCounter");
for (var count = 1; count <= numGuests; ++count) {
        Response.Write(Application.Contents("guest" +
        count) + "<BR>");
}
Application.Unlock();
%><HR>
```

5．输入下面的表单，它包含了惟一一个按钮来执行 history.back()方法返回到 HomePage.asp。

```
<FORM>
<INPUT TYPE="button" VALUE="Return to WebAdventure's
Home Page"
        onClick="history.back();">
</FORM>
```

6．添加下面的代码来结束<BODY>和<HTML>标签。

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

7．在 Data Disk 的 Tutorial.10 文件夹内将文件存为 ShowGuestBook.asp，接着将文件拷贝和上传到 ASP 服务器上。

8．在 Internet Explorer 中打开来自 ASP Web 服务器的 StartPage.asp。输入名并单击“Continue”按钮打开 HomePage.asp。填充姓和 E-mail 地址，接着单击“I’m Finished”按钮来签署客户簿。图 10-24 显示了在 Internet Explorer 中的 SignGuestBook.asp。

图 10-24　Internet Explorer 中的 SignGuestBook.asp

9．单击“ Return to WebAdventure’s Home Page ”按钮返回到 HomePage.asp，接着单击“ See Who’s Been Here ”按钮来查看已在客户簿上签名的用户清单。图 10-25 显示了在数个客户签署了客户簿之后的 ShowGuestBook.asp。

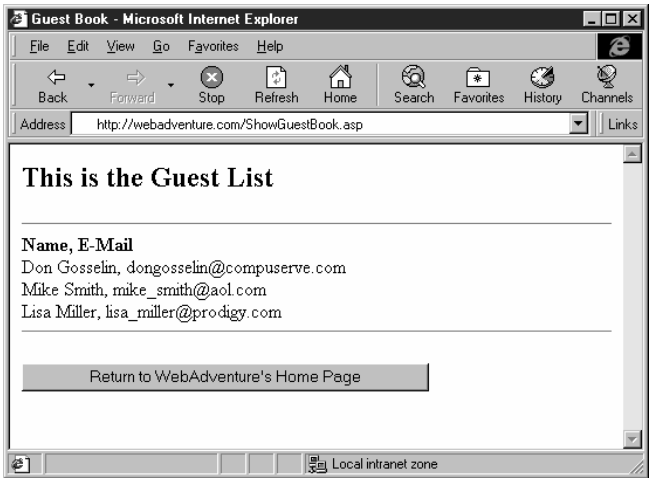


图 10-25　Internet Explorer 中的 ShowGuestBook.asp

帮助：可能会看到一条警告消息，告诉页面已经过期，它取决于 Internet Explorer 的版本。单击刷新按钮返回到 WebAdventure 主页。

10．关闭 Web 浏览器窗口。

10.2.6　总结

- ✧ Microsoft 的服务器端 JavaScript 是 Active Server Pages（ASP），它是 Microsoft Web 服务器的内置特性。

- ASP 自动识别对 ASP 应用程序的修改，在下次客户请求它时自动重新编译应用程序。
- 使用.asp 后缀的文件来创建 ASP 应用程序。
- 术语 ASP 应用程序指在同一个根目录下相关的 ASP 文件的集合。
- ASP 使用<%和%>脚本界定符来指明服务器端 JavaScript 代码。界定符是用来标识代码段开始和结束的一个字符或字符序列。
- RUNAT=SERVER 属性强制<SCRIPT>…<SCRIPT>标签对在服务器上运行。
- 使用 ASP 命令来指明在 ASP 文件中所用的脚本语言，将输出传递给浏览器。
- ASP 处理命令为 Web 服务器提供了怎样处理 ASP 文档中的脚本的信息。
- 输出命令将表达式的结果传递给 Web 浏览器。
- 可以将 HTML 标签和文本作为服务器端 JavaScript 选择结构的一部分，如 if…else 语句。
- 集合是用来保存 ASP 核心对象中的变量的数据结构，类似于数组。
- Application 对象和 Session 对象都包含了各自的 Contents 集合，它们用来保存自定义变量。
- count 属性返回集合中变量的数目。
- ASP 核心对象是 Request、Response、Session、Application 和 Server。
- ASP Request 对象代表来自客户的当前 URL 请求，与 LiveWire 的 Request 对象等价。每次客户请求一个 URL 时，ASP 都创建一个新的 Request 对象。
- Response 对象将输出和信息返回给客户。
- Session 对象用来临时保存特定的客户信息，它能够被 ASP 应用程序中的所有页面访问。ASP Session 对象与 LiveWire Client 对象等价。在客户首次访问给定 ASP 应用程序中的 URL 时将初始化一个 Session 对象。
- Application 对象被用来保存应用程序的全局信息，并且能够被所有访问应用程序的客户共享。ASP Application 对象与 LiveWire 的 Project 对象等价。
- 每个应用程序具有自己的 Application 对象。
- ASP 应用程序在客户首次请求应用程序的一个页面时自动启动。
- Lock()方法防止其他客户访问 Application 对象的属性，Unlock()方法取消 Lock()方法。
- Server 对象让 ASP 的应用程序能够访问能够被服务器上的所有应用程序访问的属性和方法。

10.2.7　问题

1．使用____标签将服务器端 JavaScript 代码添加到 ASP 应用程序中的 HTML 文档中。

a．<ASP>…</ASP>

b．<%…%>

c．<SERVERSCRIPT>…</SERVERSCRIPT>
d．<ASPSERVER>…<ASPSERVER>
2．____属性强制<SCRIPT>…</SCRIPT>标签对在服务器上运行。
a．RUNAT=SERVER
b．SERVER=TRUE
c．SERVERSIDE
d．SERVEREXECUTE
3．哪种用来创建将默认的脚本语言设置为 JScript 的处理命令的语法正确？
a．<% LANG=JScript %>
b．<%@ DEFAULT=Jscript %>
c．<% LANGUAGE=JScript %>
d．<%@ LANGUAGE=JScript %>
4．处理命令必须放置在 HTML 文件中的何处？
a．在第一个 ASP 脚本段中
b．在<HEAD>…</HEAD>标签内
c．在<HTML>标签前
d．在<HTML>标签后
5．在 HTML 标签内包括 ASP 的 Write()方法的正确语法是哪一条？
a．<% Write("Hello World"); %>
b．<%= Write("Hello World"); %>
c．<% Response.Write("Hello World"); %>
d．<%= Response.Write("Hello World"); %>
6．怎样编译 ASP 应用程序？
a．使用 jcompile
b．使用 jsac
c．使用 aspcomp
d．通过打开应用程序中的文件
7．Application 和 Session 对象中具有哪种集合？
a．Contents
b．Preferences
c．Cookies
d．Properties
8．哪种对象包含了 Form 集合？
a．Request
b．Response
c．Session
d．Application

9．下面哪种对象生存期最短？

a．Request

b．Response

c．Session

d．Application

10．怎样使用 Request 对象引用“ password ”表单域？

a．Request.password

b．Request.Form.password

c．Request.Form{"password"}

d．Request.Contents("password")

11．使用哪种方法来改变 Session 对象的生存期？

a．timeout()方法

b．Timeout 属性

c．lifespan 属性

d．Duration()方法

12．在哪种对象中保存递增的、必须能够被访问 ASP 应用程序的所有客户访问的订单号？

a．Request

b．Session

c．Application

d．Server

13．哪种方法防止其他客户访问 Application 对象或 Server 对象的属性？

a．lock()

b．Lock()

c．frozen()

d．Preserve()

10.2.8　练习

1．创建一个 ASP 猜数游戏，类似于在第 9 章中创建的随机数游戏。在 Session 对象中记录每个客户所猜的数。还在 Application 对象中记录用户名和高分。还要允许用户查看姓名和高分清单。

2．European Computer Manufacture’s Association（ECMA）制定了一个国际化的、标准版的客户端 JavaScript，称为 ECMAScript。没有与服务器端 JavaScript 相对应的标准。为什么会这样，是否正在制定服务器端 JavaScript 标准，搜索 Internet 和参阅 LiveWire 和 Microsoft 的 ASP 文档，查找与它相关的内容，接着基于所查找的内容撰写一篇报告。

3．客户端和服务器端 JavaScript 支持哪些共同的功能和对象，根据分析撰写一页使用

LiveWire 和 ASP 的报告。

4．创建一个通用的重定向页面，可以利用它向用户传递不是他们所请求的页面。将此页添加到 JavaScript 工具库中。请使用 ASP Response 对象的 Redirect()方法。

5．Microsoft 的脚本调试器工具用来处理 ASP 调试。参阅脚本调试文档并撰写一篇短论文，它描述了调试 ASP 应用程序的技术和这些技术与调试客户端 JavaScript 不同的地方。

第 11 章　数据库连接

案例

WebAdventure 正在启动一个培训计划，介绍各种 Web 开发科目，包括使用 JavaScript、创建用于 Web 的小应用程序、使用图形化的 HTML 编辑器等。WebAdventure 的经理可能要求创建一个在线注册程序，在服务器数据库中保存学生和课程登记信息。为了创建注册程序，需要学习怎样使用 LiveWire 和 Active Server Pages（ASP）来访问数据库。

浏览注册程序

在本教程中，将使用 LiveWire 和 ASP 为 WebAdventure 计算机课程创建一个在线注册程序。注意本教程建立在第 10 章所讲的 LiveWire 和 ASP 版的服务器端 JavaScript 的基础上。在开始本教程之前，请一定完全理解 LiveWire 和 ASP 的基本概念。

注册程序将与数据库协同运行。LiveWire 和 ASP 具有不同的数据库访问机制。在 11.1 节，将学习怎样使用 LiveWire 访问数据库；在 11.2 节，将学习怎样使用 ASP 访问数据库。

可以将注册程序想象为一种类型的在线“购物车”，能够对它进行修改用于电子商务（e-commerce）应用程序。图 11-1 显示了在 Navigator 中注册程序首页的样式。

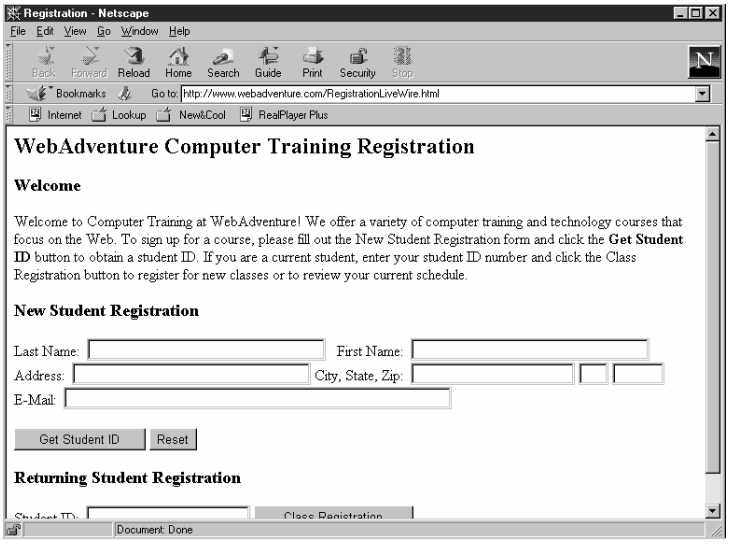


图 11-1　注册程序

提示：真正的能够接收信用卡付账的电子商务应用程序安装在安全的服务器上，并且使用了许多先进的安全协议。在本教程中的注册程序仅介绍了基本的、怎样使用服务器端 JavaScript 从数据库读取和写入内容。要查阅有关电子商务安全和策略的内容，请在 Netscape 网址 DevEdge Online http://developer.netscape.com 和 Microsoft Developer Network http://msdn.mircrosoft.com/ 查找关键词 commerce 或 e-commerce。

11.1 数据库概要和使用 LiveWire 连接数据库

本节目标

在本节将学习：

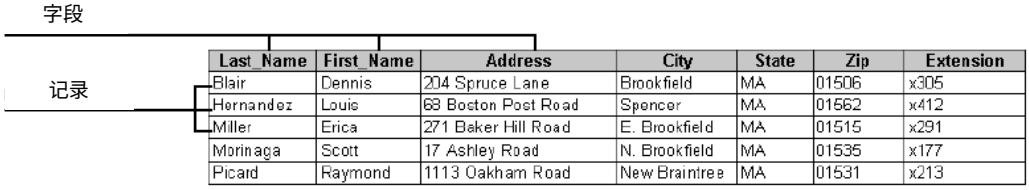
- ✧ 数据库基本架构
- ✧ 数据库管理系统
- ✧ 结构化查询语言
- ✧ LiveWire 的 Database 对象
- ✧ 怎样使用 LiveWire 执行 SQL 命令
- ✧ 怎样使用 LiveWire 执行事务处理
- ✧ 怎样使用 LiveWire 处理数据库错误

11.1.1 理解数据库

本教程的目标是学习怎样使用服务器端 JavaScript（LiveWire 和 ASP）来读取、写入和修改数据库信息。为了实现此目标，理解数据库的工作机理是有帮助的。数据库的标准定义是一组有序的信息的集合，计算机能够快速地从中访问信息。可能想到许多日常生活中的数据库。例如，地址簿是一个数据库；厨房中介绍菜谱的卡片也是一个数据库；数据库的其他实例包括企业的雇员目录和装有客户信息的文件柜。实质上，任何能够被组织为有序数据集、接着被快速检索的信息都可以看作是一个数据库。数百个装在鞋盒中棒球卡片集合不是一个数据库，因为不能够快速或方便地检索每一张纸牌（除非很幸运）。然而，若按照球队将棒球卡片集合在活页夹中组织，接着更进一步地根据每位球员的位置或击球平均次数组织，那么就可以将它想象是一个数据库，因为能够十分迅速地查找到指定的卡片。

存储在计算机数据库中的信息实际上存储在类似于电子表格的数据库表中。数据库表中的每一行称为一条记录。数据库中的一条记录是一组单一的、完整的、相关信息的集合。例如，菜谱数据库中的每种菜谱都是一条数据库记录。存储在记录中的每个独立的信息片断称为字段。菜谱数据库中存在的字段实例包括成分、烹饪时间、烹饪温度等。

总而言之，可以将数据库看作由表组成，它又由记录组成，记录又由字段组成。图 11-2 显示了 WebAdventure 内程序员的通讯录实例。数据库由 5 条记录组成，每个雇员对应一条记录。每条记录由 7 个字段组成：Last_Name、First_Name、Address、City、State、Zip 和 Extension。



Last_Name	First_Name	Address	City	State	Zip	Extension
Blair	Dennis	204 Spruce Lane	Brookfield	MA	01506	x305
Hernandez	Louis	68 Boston Post Road	Spencer	MA	01562	x412
Miller	Erica	271 Baker Hill Road	E. Brookfield	MA	01515	x291
Morinaga	Scott	17 Ashley Road	N. Brookfield	MA	01535	x177
Picard	Raymond	1113 Oakham Road	New Braintree	MA	01531	x213

图 11-2　雇员通讯录数据库

图 11-2 中的数据库是一个平面文件数据库实例，它属于数据库最简单类型中的一种。平面文件数据库将信息存储在一张表中。对十分简单的信息集合，平面文件数据库通常就足够了。对大型的复杂的信息集合，平面文件数据库就不适用了。对大型的复杂的数据库更好的解决方法是关系数据库。关系数据库跨多张相关的表存储信息。尽管在本教程中，并不真正使用关系数据库，但是理解它们的工作原理十分有益，因为关系数据库是现在使用的最普遍的数据库。

提示：可能遇到的另外两种数据库系统类型是层次数据库和网络数据库。

关系数据库由一张或多张相关联的表组成。实际上，大型的关系数据库可能由几十张甚至数百张相关联的表组成。尽管关系数据库可能由许多表组成，但是可以在数据库中创建关系来一次处理两张相关的表。 一种关系中的一张表总可以看作是主表，同时另外一张表可以看作是相关联的表。主表是关系中被另外一个表引用的主要的表。相关表或子表引用关系数据库中的主表。关系中的表通过主键和外键关联起来。主键是主表中一个字段，它用来惟一地标识主表中的每条记录。外键是相关表中的一个字段，它用来引用主表中的主键。主键和外键用来链接关系数据库中多个表内的记录。

在关系数据库中存在三种基本的关系类型：一对一、一对多和多对多。对主表中的每条记录，相关表中的每条记录仅含有惟一一条记录，这时在两个表之间存在一对一关系。在将信息分割成多个逻辑集合时，可以创建一对一关系。理解存储在一对一关系内表中的信息还能够存储在一张表中十分重要。然而，可能想将信息分割存储在多张表中以便更好地将信息组织成逻辑集合。使用一对一关系的另一个原因是：其中的一个表是机密的并且仅能被指定的人员访问。例如，可能想创建职员表来存储与雇员相关的基本信息，类似于图 11-2 中表的内容。可能还想创建工资表，它用来存储每个雇员薪水、福利、其他种类的补偿以及仅能够被人事科和财务科访问的机密信息。图 11-3 显示了两个表：Employees 和 Payroll，它们具有一对一关系。主表是图 11-2 中的职员信息表。相关表是工资表，它存储机密的薪水和补偿信息。请注意每个表都含有相同的记录号，在主表中的一条记录与相关表中的一条记录相对应。通过为 Employees 表添加主键和为 Payroll 表添加外键来创建一对一关系。

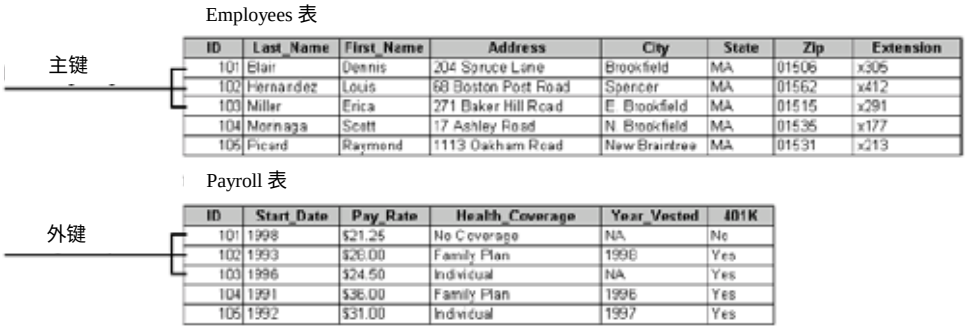


图 11-3 一对一关系

在主表中的每条记录对应相关表中多条记录时，那么在关系数据库中就存在一对多关系。主键和外键是关系数据库表中惟一需要重复的信息。为了消除在单个表中的冗余信息，可以创建一对多关系。为了减少冗余和重复信息而将表分割成多个相关的表的方法称为规范化（Normalization）。消除冗余信息（规范化）减少了数据库的尺寸并且使数据更加容易管理。例如，考虑图 11-4 中的表。它列出了 WebAdventure 每个程序员使用的主要的和次要的编程语言。注意每个程序员的姓名重复出现在每一种他或她所熟悉的编程语言的记录中。上述的重复是冗余信息的一个实例，它通常发生在单个表内。

ID	Last_Name	First_Name	Programming_Language
101	Blair	Dennis	Client-Side JavaScript
101	Blair	Dennis	Server-Side JavaScript
102	Hernandez	Louis	Client-Side JavaScript
102	Hernandez	Louis	Server-Side JavaScript
102	Hernandez	Louis	CGI
103	Miller	Erica	Client-Side JavaScript
103	Miller	Erica	Server-Side JavaScript
103	Miller	Erica	CGI
103	Miller	Erica	Perl
104	Morinaga	Scott	Client-Side JavaScript
104	Morinaga	Scott	CGI
104	Morinaga	Scott	Perl
105	Picard	Raymond	Client-Side JavaScript
105	Picard	Raymond	CGI

图 11-4 具有冗余信息的表

一对多关系为在数据库中存储此类信息提供了一种更有效和更少冗余的方法。图 11-5 显示了对相同内容通过一对多关系组织后的形式。

提示：一对多关系中的“多”的一方有时被用作主表。在这些案例中，关系通常被称为多对一关系。

尽管图 11-5 是一对多关系的一个实例，但是表没有经过规范化，因为在 Programming_Language 域中包含了重复的值。请记住主键和外键是关系数据库中惟一可以重复的信息。为了更进一步地减少冗余，可以将图 11-5 中的“多”表组织为另外一个一对多关系。然而，更好的选择是多对多关系。当一个表中的多条记录与另一个表中的多条记

录关联时，在关系数据库中就存在了多对多关系（Many-to-Many Relationship）。请考虑程序员和编程语言之间的关系。每位程序员能够使用多种编程语言，同时每种编程语言能够被多位程序员使用。为了创建多对多的关系，必须使用连接表，因为多数的关系数据库系统不能够直接处理多对多关系。连接表（Junction Table）为多对多关系中的每个表都创建了一个一对多关系。连接表含有来自多对多关系中两个表的外键，同时还含有任何其他与多对多关系相应的域。图 11-6 包含了 Programmers 表和 Programming Languages 表之间多对多关系的一个实例。Programmers 表包含了 Programmer_ID 主键，而在 Programming Languages 表中包含了 Language_ID 主键。Programming Experience 连接表包含了两个外键，它们分别与 Programmers 表中的 Programmer_ID 主键和 Programming Languages 表中的 Language_ID 对应。Programming Experience 连接表还包含了 Years_Experience 域。可以向 Programming Experience 连接表中添加记录来创建每个程序员已使用某种特殊编程语言的年数列表。

“一”方

ID	Last_Name	First_Name
101	Blair	Dennis
102	Hernandez	Louis
103	Miller	Erica
104	Morinaga	Scott
105	Picard	Raymond

“多”方

ID	Programming_Language
101	Client-Side JavaScript
101	Server-Side JavaScript
102	Client-Side JavaScript
102	Server-Side JavaScript
102	CGI
103	Client-Side JavaScript
103	Server-Side JavaScript
103	CGI
103	Perl
104	Client-Side JavaScript
104	CGI
104	Perl
105	Client-Side JavaScript
105	CGI

图 11-5　一对多关系

Programmers 表

Programmer_ID	Last_Name	First_Name
101	Blair	Dennis
102	Hernandez	Louis
103	Miller	Erica
104	Morinaga	Scott
105	Picard	Raymond

Programming Languages 表

Language_ID	Programming_Language
10	Client-Side JavaScript
11	Server-Side JavaScript
12	CGI
13	Perl

Programming Experience 连接表

Programmer_ID	Language_ID	Years_Experience
101	10	5
101	11	4
102	10	3
102	11	2
102	12	3
103	10	2
104	12	3
104	13	5
105	11	3

图 11-6　多对多关系

11.1.2 数据库管理系统

掌握了数据库的基本设计之后，现在可以开始考虑怎样创建和管理数据库了。创建、访问和管理数据库的一个应用程序或多个应用程序集被称为数据库管理系统（Database Management System，或 DBMS）。数据库管理系统可以在多种不同的平台上运行，从个人计算机到客户/服务器系统，乃至大型机。不同类型的数据库格式具有不同的数据库管理系统。以平面文件格式存储数据的数据库管理系统称为平面文件数据库管理系统。以关系格式存储数据的数据库管理系统称为关系数据库管理系统（或 RDBMS）。其他类型的数据库管理系统包括层次和网络数据库管理系统。已听说的某些流行的关系数据库管理系统包括用于大型机的 Oracle、Sybase、Informix 和 DB2，以及用于 PC 的 Access、Foxpro 和 Paradox。

数据库管理系统执行正在使用的其他类型的应用程序的大部分相同的功能，如字处理和电子表格应用程序。例如，数据库管理系统创建了新的数据库文件并且提供了让用户输入和管理数据的接口。数据库管理系统最重要的功能之一是结构化和保存数据库文件的结构。另外，数据库管理系统必须保证将数据正确地保存在数据库表中，而不管数据库格式（平面文件、关系性、层次型或网络）。例如，在关系数据库中，数据库管理系统确保根据数据库表中的关系输入合适的信息。许多 DBMS 系统还具有安全特性，它用来限制用户访问特殊类型的数据。

数据库管理系统的其他两个重要的方面是它们的查询和报表功能。查询是用来检索、添加、修改和删除数据库信息的结构化的命令和规则集。报表是数据库表或查询结果的格式化打印输出。多数关系数据库系统使用数据操纵语言（DML）来创建查询。不同数据库管理系统支持不同的数据操纵语言。然而，结构化查询语言（SQL，念作 sequel）已成为众多数据库管理系统之间的标准数据操纵语言。

众多数据库系统将数据操纵语言隐藏在用户接口后，让用户创建查询更容易。图 11-7 显示了 Access 的查询设计界面的一个实例。用户能够将屏幕上部分表对象中的字段拖入屏幕下部分的规则网格中来创建查询。在后台，Access 将创建图 11-8 中所示的 SQL 代码。SQL 是 Access 的数据操纵语言。

尽管使用接口来设计查询对终端用户十分友好，但是若通过编程方式来维护数据库中的数据，那么必须学习数据库管理系统的数据操纵语言。例如，在使用服务器端 JavaScript 访问数据库时，必须使用数据操纵语言。因为 SQL 是众多数据库管理系统底层的数据操纵语言，所以在本节的后面部分将学习更多的与 SQL 相关的知识以便能够使用服务器端 JavaScript 和数据库。

提示：众多数据库管理系统还使用数据定义语言（DDL）来创建数据库、表和数据库的其他组件。

尽管众多数据库管理系统支持相同的数据库格式（平面文件、关系、层次或网络），

但是每个数据库管理系统是一个独立的应用程序，它们创建自己所有的文件类型，理解这一点十分重要。例如，尽管 Access 和 Paradox 都是关系型数据库管理系统，但是 Access 以自己特有的格式来创建后缀为.mdb 数据库文件；而 Paradox 使用自己特有的格式来创建后缀为.db 的数据库文件。尽管 Paradox 和 Access 都包含了过滤器让导入彼此的文件格式，但是在这两种程序之间数据库文件不能完全互换。这种情形对众多的数据库管理系统经常发生。它们能够直接导入彼此的文件格式，但是它们不能直接读取彼此的文件。数据库管理系统的私有特性意味着创建访问特殊数据库管理系统的文件格式的 JavaScript 应用程序时，必须编写适合于那种数据库管理系统的应用程序。

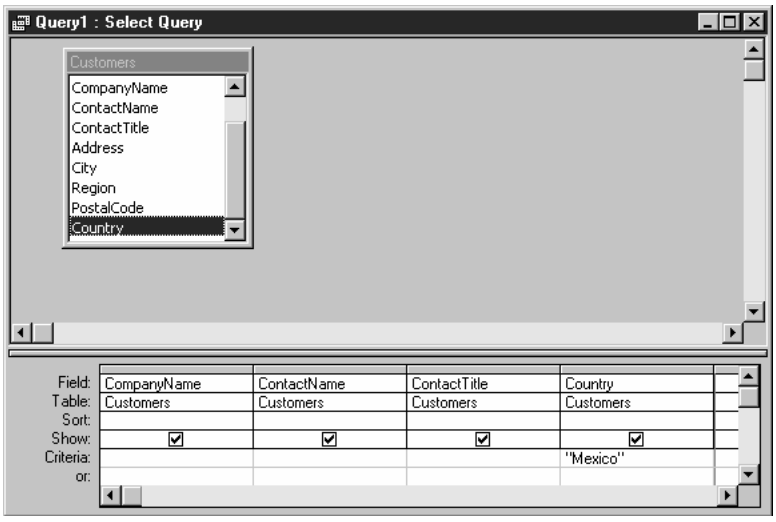


图 11-7　Access 查询设计界面

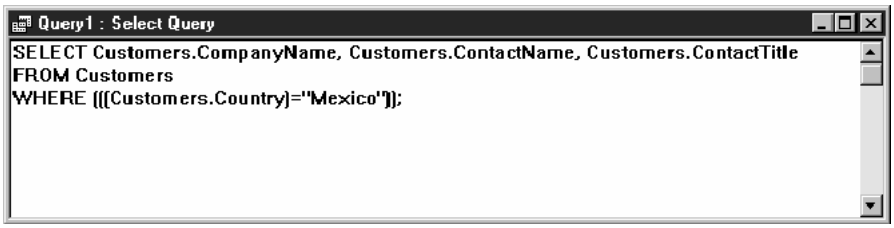


图 11-8　Access SQL 代码

在目前的环境中，通常一个应用程序访问多个不同数据库管理系统中创建的数据库是必要的。例如，某企业可能需要一个服务器端 JavaScript 应用程序，它能够同时访问一个大型的传统的用 dBase 编写的数据库以及一个新的用 Oracle 编写的数据库。将大型 dBase 数据库转换成 Oracle 受成本限制。另一方面，企业不能继续使用旧的 dBase 数据库，因为它的需要已经超越了旧的数据库的能力。企业还必须能够访问两种系统中的数据。为了方便访问各种数据库格式中的数据，Microsoft 提出了开放数据库连接标准。开放数据库连接（ODBC）让所编写的遵守标准的应用程序能够访问任何数据源，与之相应地还有一个

ODBC 驱动程序。ODBC 使用 SQL 命令（众所周知的 ODBC SQL）让与 ODBC 兼容的应用程序能够访问数据库。实质上，ODBC 应用程序连接到数据库（与之相应地存在着一个 ODBC 驱动程序），然后执行 ODBC SQL 命令。接着，ODBC 驱动程序将 SQL 命令译成一种数据库能够理解的格式。LiveWire 和 ASP 都是与 ODBC 兼容的应用程序，它们能够访问任何数据库（与之相应地存在着一个 ODBC 驱动程序）。

提示：LiveWire 还直接访问 Informix、Oracle、Sybase 和 DB2 数据库的源数据文件格式。在 LiveWire 中使用源文件格式通常比使用 ODBC 快，因为应用程序不必经过整个额外的驱动程序层。然而，存在一种协定，在使用源数据库格式时，必须知道每种数据库管理系统的数据操纵语言。

在本教程中，将使用已有的 Microsoft Access 数据库 WebAdventureCourse.mdb。既然 Access 数据库是与 ODBC 兼容的，所以它们能够被 LiveWire 和 ASP 使用。WebAdventureCourse.mdb 数据库由两个表组成：Students 和 Registration。Students 表将包含每位学生的 ID 和姓名，以及其他个人信息。Registration 表将为学生所参加的每个班都包含一条记录。Students 表是主表，Student_ID 字段作为主键。Student_ID 字段还作为 Registration 表中的外键。既然每个学生能够参加多个班，那么 Students 表和 Registration 表之间的关系是一对多关系；Students 表是关系中的“一”的一方，而 Registration 表是关系中“多”的一方。

为了使访问 32 位 Windows 操作系统上的与 ODBC 兼容的数据库更加容易，将创建一个数据源名来定位和标识数据库。数据源名（DSN）含有配置信息，Windows 操作系统使用它来访问特殊的与 ODBC 兼容的数据库。若在 Windows 平台上安装了 Netscape Enterprise 或 FastTrack 服务器，那么必须建立 DSN 来执行本节中的 LiveWire 练习。还必须建立 DSN 来执行 11.2 节中的 ASP 练习。

提示：若除了在 Windows 操作系统上还在其他操作系统上安装了 Enterprise 或 FastTrack 服务器，那么请参考 Netscape 的 LiveWire 文档（位于 http://developer.netscape.com/）来为特殊的平台了解连接与 ODBC 兼容的数据库相关的信息。

在 Windows 环境下，使用控制面板中的 ODBC 管理器应用程序来安装和管理所能够连接的 DSN。为了让 LiveWire 访问 Windows 环境下的数据源，必须使用 3.5 版的 ODBC 管理器，它包括了最新的用于 ODBC 数据源的驱动程序，包括 Access 97。从 Microsoft 网址下载 WX1350.exe，能够在 Windows 上安装 3.5 版的 ODBC 管理器。

有三种类型的数据源：系统、用户和文件。系统 DSN 让所有登录到服务器的用户都能够访问数据库。用户 DSN 仅让授权的用户访问。文件 DSN 创建了一个基于文件的、后缀为.dsn 的数据源，它能够在用户之间共享。在本教程中将创建系统 DSN。对 LiveWire 程序，将创建 WebAdventureLiveWire 系统 DSN。

提示：LiveWire 不能够使用文件 DSN。

创建用于 LiveWire 数据库程序的系统 DSN：

1．单击 Windows 的开始菜单，接着选中设置文件夹下的控制面板。

2．单击或双击（取决于是怎样配置桌面的）控制面板窗口中的 ODBC 图标。

帮助：控制面板中的 ODBC 图标可能具有不同的标题（取决于 Windows 的版本）。例如，在 Windows 98 下，ODBC 的标题为 ODBC 数据源（32 位）。

3．在 ODBC 数据源管理器窗口中，选中系统 DSN 标签，接着单击“添加”按钮。在弹出的窗口内，选中 Microsoft Access Driver（*.mdb）并单击“完成”按钮。

4．在 ODBC Microsoft Access 97 安装窗口内，输入 WebAdventureLiveWire 作为新 DSN 文件的名称，接着单击“选择”按钮。在弹出的选择数据库对话框内，选中 Data Disk 的 Tutorial.11 文件夹下的 WebAdventureCourse.mdb 文件并单击“确定”。接着将关闭选择数据库对话框。

5．单击“确定”关闭 Microsoft Access 安装对话框，接着单击“确定”关闭 ODBC 数据源管理器窗口。最后，关闭控制面板。

下一步，将创建主 RegistrationLiveWire.html 文档。在学生访问 WebAdventure 的注册网址时，RegistrationLiveWire.html 文件将是学生看到的第一个 Web 页。此文档仅含有 HTML 文本和表单，未包含任何客户端或服务器端 JavaScript 代码。RegistrationLiveWire.html 将使用每个表单的提交按钮调用访问数据库的 LiveWire 文档。

创建主 RegistrationLiveWire.html 文档：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．输入文档的起始<HTML>、<HEAD>和<BODY>段。

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

3．添加下面的组成文档的标签、文本和表单。两个表单都使用提交按钮来调用 LiveWire 文档。第一个表单调用 GetStudentIDLiveWire.html LiveWire 文档，它为学生 ID 赋值。第二个表单调用 CourseListingLiveWire.html 文档，现有学生使用它来注册新的班级或回顾他们当前的课程表。在本节的后面部分，将创建这些提交按钮调用的 LiveWire 文档。

```
<H2>WebAdventure Computer Training Registration</H2>
<H3>Welcome</H3>
<P>Welcome to Computer Training at WebAdventure! We offer
a variety of computer training and technology courses
that focus on the Web. To sign up for a course, please
fill out the New Student Registration form and click the
<B>Get Student ID</B> button to obtain a student ID.If
```

```
you are a current student, enter your student ID number
and click the Class Registration button to register for
new classes or to review your current schedule.</P>
<H3>New Student Registration</H3>
<FORM NAME="customerInfo" METHOD="post"
ACTION="GetStudentIDLiveWire.html">
Last Name:  <INPUT TYPE="text" NAME="last_name"
SIZE=30>  
First Name:  <INPUT TYPE="text" NAME="first_name"
SIZE=30><BR>
Address:  <INPUT TYPE="text" NAME="address" SIZE=30>
City, State, Zip:  <INPUT TYPE="text"
NAME="city" SIZE=20>
<INPUT TYPE="text" NAME="state" SIZE=2 MAXLENGTH=2>
<INPUT TYPE="text" NAME="zip" SIZE=5 MAXLENGTH=5><BR>
E-Mail:  <INPUT TYPE="text" NAME="email" SIZE=50><P>
<INPUT TYPE="submit" NAME="submit ”
VALUE=" Get Student ID ">
<INPUT TYPE="reset"><P>
</FORM>
<H3>Returning Student Registration</H3>
<FORM METHOD="post" ACTION="CourseListingLiveWire.html">
Student ID:  <INPUT TYPE="text" NAME="id">
<INPUT TYPE="submit" VALUE=" Class Registration ">
</FORM>
```

4．添加下面的代码结束<BODY>和<HTML>标签。

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

5．在 Data Disk 的 Tutorial.11 文件夹内将文件存为 RegistrationLiveWire.html。在 Navigator 中打开 RegistrationLiveWire.html。在本节后面部分，将 RegistrationLiveWire.html 和其他组成注册程序的文件编译成 LiveWire 应用程序。图 11-9 显示了文档在 Navigator 中的显示结果。在执行表单按钮之前，需要创建此 LiveWire 文档。

6．关闭 Web 浏览器窗口。

11.1.3 结构化查询语言

在 20 世纪 70 年代 IBM 根据某种规则发明了 SQL 作为查询数据库的一种手段。从那

时起运行在大型机、微型机和 PC 上的数据库管理系统都对它进行了改编。在 1986，美国国家标准委员会（ANSI）批准 SQL 语言为正式的标准。在 1991 年，X/Open 和 SQL Access Group 创建了 SQL 的一个标准化的版本，它就是众所周知的公共应用程序环境（CAE）。尽管有两种主要的标准可用，然而，众多的数据库管理系统使用它们自己的 SQL 语言。ODBC SQL 对应于 X/Open 和 SQL Access Group 的 CAE SQL 草案标准。因此，特定数据库管理系统的 ODBC 驱动程序必须支持 ODBC SQL。

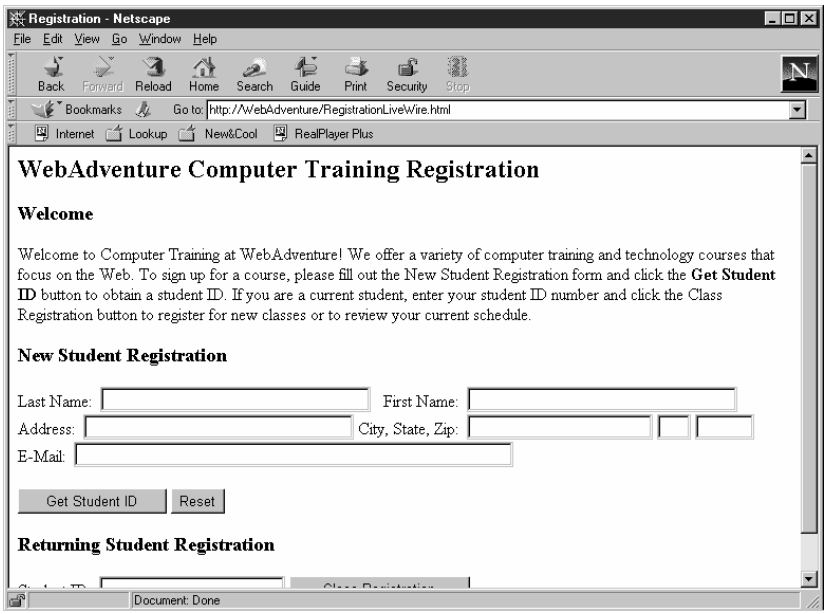


图 11-9　Navigator 中 RegistrationLiveWire.html

提示：若曾经直接使用过数据库管理系统，那么请记住在本教程中所学的 ODBC SQL 可能不直接与那种数据库管理系统的 SQL 相对应。

SQL 使用十分容易理解的语句来执行数据库命令。SQL 命令由对数据库进行操作的关键字组成。表 11-1 列出了几种对多数 SQL 都通用的 SQL 关键字。

表 11-1　通用 SQL 关键字

关　键　字	描　　述
FROM	指定所要检索或删除记录的表
SELECT	从数据库中检索信息
WHERE	指定查询所返回的记录必须满足的条件
ORDER BY	对数据库检索的结果进行排序
INSERT	往数据库中插入新行
INTO	确定记录所要插入的表
DELETE	从数据库中删除记录
UPDATE	保存对记录字段的修改

SQL 语句 SELECT * FROM Programmers 从 Programmers 表中选中所有记录（使用星号*）。下面的代码显示了一个更复杂的 SQL 语句，它从 Programmers 表中选中 City 字段值等于 Spencer 的记录的 Last_Name 和 First_Name 字段。接着对 Last_Name 和 First_Name 字段应用 ORDER BY 关键字对结果排序。

```
SELECT Last_Name, First_Name FROM Programmers
WHERE City = "Spencer" ORDER BY Last_Name, First_Name
```

在将 ODBC SQL 与服务器端 JavaScript 一起使用时，SQL 语句将作为文本字符串创建，它作为 LiverWire 和 ASP 方法的参数使用。例如，LiveWire 的 Database 对象包括了一个执行 SQL 字符串的 SQLTable 方法。在 LiveWire 执行前面的代码，请使用下面的语句。第一条语句将 SQL 字符串赋给 SQLString 变量，第二条语句使用 SQLTable 方法执行此语句。

```
var SQLString =
"SELECT Last_Name, First_Name FROM Programmers
      WHERE City = 'Spencer' ORDER BY Last_Name, First_Name";
database.SQLTable(SQLString);
```

在本教程中将学习怎样使用几个基本的 ODBC SQL 关键字。要深入地了解 ODBC SQL，请访问 MicroSoft Developer Network http://msdn.microsoft.com/。

在学习怎样使用 LiveWire 访问数据库之前，需要创建一个 CourseListingLiveWire.html HTML 文档。WebAdventure 的计算机培训班的学生将使用 CourseListingLiveWire.html 文件来选择他们所想参加的课程和回顾当前的课程表。CourseListingLiveWire.html 由两个表单组成。第一个表单让学生回顾当前的课程表，第二个表单让学生注册新课程。每个表单的提交按钮所调用的 LiveWire 文档都包含了必要的从注册数据库中读取和写入内容的功能。

创建 CourseListingLiveWire.html 文件：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．输入起始<HTML>、<HEAD>和<TITLE>标签。

```
<HTML>
<HEAD>
<TITLE>Home Page</TITLE>
```

输入下面的<SERVER>段。if 语句使用逻辑非（!）运算符来判断在 Client 对象中是否存在 sameSession 对象。若不存在 sameSession 对象，将创建它并将它赋为 true。接着将 Request 对象的 ID 属性赋给 Client 对象的 studentID 属性。Request 对象的 ID 属性包含了学生输入 Registration 表单中 ID 域的内容。在用户浏览组成程序的页面时，使用 studentID 属性在整个注册程序中跟踪用户。

```
<SERVER>
client.studentID = request.id;
if (!client.sameSession) {
        client.sameSession = "true";
client.studentID=request.id;
}
</SERVER>
```

帮助：Client 对象临时存储特殊的、能够让 LiveWire 应用程序中的所有页面访问的客户信息。Request 对象接收客户浏览器内表单中的所有命名元素并将它们作为 Request 对象的属性添加。在第 10 章中首次使用了 Client 和 Request 对象。

3．添加</HEAD>标签。

4．添加下面的 HTML 标签、文本和表单。该表单包含了两个用于显示学生课程表的元素。使用<SERVER>…</SERVER>标签对将学生的 ID 打印在表单内的屏幕上。

```
<BODY>
<H3>Course Registration Form</H3>
<FORM METHOD="post" ACTION="ReviewScheduleLiveWire.html">
<B>Student ID:  <SERVER> write(client.studentID)
</SERVER></B>
<INPUT TYPE="submit" VALUE=" Review Current Schedule "><P>
</FORM>
```

5．输入下一个表单，学生使用它来注册课程。表单的内容通过 RegisterStudent LiveWire.html 文件提交给 LiveWire。

```
<FORM METHOD="post" ACTION="RegisterStudentLiveWire.html">
<B>Select the course you would like to take:</B><BR>
<INPUT TYPE="radio" NAME="course" VALUE="Introduction to
Active Server Pages 4.0">Introduction to Active Server
Pages<BR>
<INPUT TYPE="radio" NAME="course" VALUE="Introduction to
JavaScript">Introduction to JavaScript<BR>
<INPUT TYPE="radio" NAME="course" VALUE="Introduction to
LiveWire 98">Introduction to LiveWire<BR>
<INPUT TYPE="radio" NAME="course" VALUE="Intermediate
Active Server Pages 4.0">Intermediate Active Server Pages<BR>
<INPUT TYPE="radio" NAME="course" VALUE="Intermediate
JavaScript">Intermediate JavaScript<BR>
<INPUT TYPE="radio" NAME="course" VALUE="Intermediate
LiveWire 98">Intermediate LiveWire<BR>
```

```
<INPUT TYPE="radio" NAME="course" VALUE="Advanced Active
Server Pages">Advanced Active Server Pages<BR>
<INPUT TYPE="radio" NAME="course" VALUE="Advanced
JavaScript">Advanced JavaScript<BR>
<INPUT TYPE="radio" NAME="course" VALUE="Advanced
LiveWire">Advanced LiveWire<P>
<B>Available Days and Times:</B><BR>
<SELECT NAME="days">
<OPTION SELECTED VALUE="Mondays and Wednesdays">Mondays
and Wednesdays
<OPTION VALUE="Tuesdays and Thursdays">Tuesdays and Thursdays
<OPTION VALUE="Wednesdays and Fridays">Wednesdays and Fridays
</SELECT>
<SELECT NAME="time">
<OPTION SELECTED VALUE="9 am-11 am">9 am-11 am
<OPTION VALUE="1 pm-3 pm">1 pm-3 pm
<OPTION VALUE="6 pm-8 pm">6 pm-8 pm
</SELECT><P>
<INPUT TYPE="submit" VALUE=" Register" >
<INPUT TYPE="reset">
</FORM>
```

6．添加下面的代码结束<BODY>和<HTML>标签。

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

7．在 Data Disk 的 Tutourial.11 文件夹内将文件存为 CourseListingLiveWire.html。在打开文件之前，需要编写生成新的学生 ID 的 LiveWire 脚本。

8．关闭文本编辑器和 HTML 编辑器窗口。

11.1.4 LiveWire Database 对象

LiveWire 包括了三种用来访问数据库的对象：DbPool、Connection 和 Database 对象。DbPool 和 Connection 对象用来创建和管理数据库连接池。Database 对象用来创建和管理客户和数据库之间的每个连接。本教程仅讨论使用 Database 对象的单用户数据库连接。

提示：参阅 Netscape 的 LiveWire 文档了解使用数据库连接池。

Database 对象不包括任何属性，仅具有各种用来维护和使用数据库的方法。表 11-2 列出了 Database 对象的方法。

表 11-2　Database 对象方法

方　法	描　述
beginTransaction()	开始 SQL 事务
commitTransaction()	保存当前 SQL 事务
connect()	将 LiveWire 连接至数据库
connected()	若成功连接数据库，那么返回 true
cursor()	为指定的 SQL 语句创建数据库光标
disconnect()	关闭数据库连接
execute()	向数据库管理系统发送 SQL 语句进行处理
majorErrorCode()	返回 ODBC 或数据库服务器所创建的主要的错误代码
majorErrorMessage()	返回 ODBC 或数据库服务器所创建的主要的错误消息
minorErrorCode()	返回 ODBC 或数据库服务器所创建的次要的错误代码
minorErrorMessage()	返回 ODBC 或数据库服务器所创建的次要的错误消息
rollbackTransaction()	回滚当前的 SQL 事务
SQLTable()	执行 SQL 语句并将结果作为 HTML 表返回

在 LiveWire 内使用数据库的第一步是使用 connect()方法创建与数据库的连接。connect()方法的语法是 database.connect("database type", "server name", "user name", "password", "database name");。database type 参数指定了所要访问的数据库的类型。database type 的有效参数是 DB2、ODBC、Oracle、Informix 和 Sybase。server name 指定了数据库所在的服务器的名称。对于 Windows 平台上的 ODBC 连接，请使用在控制面板内定义的 DSN 名。user name 和 password 指定了服务器上的用户和口令。database name 仅对 Informix 和 Sybase 数据库有效。对 DB2、ODBC 和 Oracle 数据库，“database name”必须为空串。在使用完数据库之后，请使用 disconnect()方法来断开连接。下面的语句是怎样连接和断开 Server1 服务器上的 Accounting Sybase 数据库的实例。

```
database.connect(
"SYBASE", "Server1", "don", "1X347", "Accounting");
additional statements ;
database.disconnect();
```

若数据库不支持 connect()方法中的某些参数，那么请为每个不支持的参数使用一个空字符串。例如，在本教程中，使用的是 WebAdventureLiveWire DSN，它是一个 ODBC 数据库。使用 ODBC 数据库，除了 database name 和 server name，不需要使用其他 connect()方法参数。因此，为了连接 WebAdventureLiveWire DSN，请使用下面的语句：

```
database.connect("ODBC", "WebAdventureLiveWire", "", "", "");
```

在试图读取、写入、添加或修改记录之前，确保已经成功地连接了数据库是一个不错的习惯。若数据库连接是有效的，那么 Database 对象的 connected()方法将返回 true 布尔值。下面的代码添加了一条 if 语句在尝试连接数据库之后检测 connected()方法。若连接是无效的，那么 LiveWire 的 write()方法将向客户返回一条消息。

```
database.connect(
"SYBASE", "Server1", "don", "1X347", "Accounting");
if (!database.connected())
write("The database is not available.");
additional statements ;
database.disconnect();
```

LiveWire 应用程序能够使用标准连接或串行连接来访问数据库。标准连接允许多个用户同时访问数据库。串行连接在某个时刻仅允许单个用户访问数据库。在同一时刻有多少个用户能够连接数据库是个重要的问题，因为许多数据库管理系统都包含一个许可证，企业根据它们所预料将访问单个数据库的并发用户的数目来购买它。标准连接通常提供更有效的性能，因为多个用户能够让他们的请求立即处理而不需要等待其他用户的请求完成。若能确定用户的最大数目不会达到数据库管理系统所允许的，那么请使用标准连接。若认为可能超出数据库管理系统所允许的最大用户数，那么请使用串行连接。通过使用在教程 10 所学的 Project 对象的 lock()和 unlock()方法来指定数据库连接是标准的还是串行的。若不使用 lock()和 unlock()方法，那么建立的是标准连接。使用 lock()和 unlock()方法建立的是串行连接。例如，前面的代码建立的是标准连接，因为它未使用 lock()和 unlock()方法。下面的代码是一个串行连接的实例：

```
project.lock();
database.connect(
"SYBASE", "Server1", "don", "1X347", "Accounting");
if (!database.connected())
        write("The database is not available.");
additional statements ;
database.disconnect();
project.unlock();
```

下一步，将开始创建 GetStudentIDLiveWire.html 文件，它将创建新的学生 ID。将含在 Project 对象内的 idNum 属性加 1 来创建新的学生 ID。接着将新值赋给 Client 对象的 studentID 属性。首次在应用程序管理器中启动注册程序时，idNum 属性的初始值为 100。请注意从 Project 对象中的属性创建新的学生 ID 并不是创建它的最好方法，因为每次重新启动注册程序时，idNum 的值都重新被初始化。（请记住每次终止应用程序或重新启动时都将销毁 Project 对象的属性）一种更好的解决方法是使用存储在数据库中的值来创建新的

学生 ID。将基于 Project 对象的 idNum 属性来创建新的学生 ID 作为重点放在此教程所介绍的概念上。

开始创建 GetStudentIDLiveWire.html 文件：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．输入起始<SERVER>标签。

帮助：既然不在文档中包括任何 HTML 标签，那么也不需要创建任何<HTML>或<BODY>标签。

3．添加下面的代码，它锁定 Project 对象并创建一个新的学生 ID。若 studentID 变量不存在，那么将创建它。若它存在，那么将当前值加 1。

```
project.lock();
if (!project.idNum) {
        project.idNum = 100;
        client.studentID = project.idNum;
}
else {
        var curID = project.idNum;
        ++curID;
        client.studentID = curID;
        project.idNum = curID;
}
```

4．输入下面部分，它打开一个与 WebAdventureLiveWire DSN 的数据库连接。

```
database.connect("ODBC", "WebAdventureLiveWire", "", "", "");
if (!database.connected())
        write("The database is not available.");
```

5．输入结束的</SERVER>标签。

6．在 Data Disk 的 Tutorial.11 文件夹内将文件存为 GetStudentIDLiveWire.html。

11.1.5　执行 SQL 命令

Database 对象的三种方法 execute()方法、cursor()方法和 SQLTable()方法被用来执行对数据库的 SQL 查询。每种方法具有不同的用法，正如将看到的。请记住一点，本教程中的实例都是根据雇员的姓来查找记录。在实际的应用环境中，应该查找惟一的标识符，如赋给记录主键的值。

execute()方法

execute()方法向数据库管理系统发送用于处理的 SQL 语句。使用 execute()方法来更新或修改数据库表。不要使用 execute()方法来提交 ODBC SQL 语句。相反，用正在访问的数据库管理系统的 SQL 语言来提交语句。通过 execute()方法执行的语句称为转移 SQL（passthrough SQL），因为它们不是直接传递给数据库管理系统。execute()方法的语法是 database.execute(SQL 字符串);。在使用 execute()方法时，确保理解了用于目标数据库管理系统的 SQL 语法，或至少将命令局限于如 SELECT、INSERT 和 DELETE 等普通的 SQL 关键字。

execute()方法不将查询结果返回给 LiveWire。因此，不能使用 execute()方法从数据库中检索数据。相反，execute()方法对往数据库中插入、更新或删除行十分有用。例如，下面的代码使用 SQL INSERT 和 INTO 子句向 Employees 数据库表中追加一条新的雇员记录。实例中的 INTO 子句紧跟着的是数据库表名，接着是 VALUES 语句，再就是括弧，它包含了将要插入新记录中去的每个字段的值。括弧中每个值的位置与字段在 Employees 表中的位置相对应。

```
var SQLString = "INSERT INTO Employees VALUES('106', 'Mbuti',
'Pierre', '106 Flagg Road', 'Spencer', 'MA', '01562', 'x413')";
database.execute(SQLString);
```

提示：INSERT 语句在数据库表的末尾追加新记录。在本节的后面部分将学习怎样使用 Cursor 对象的 insertRow()方法在表中的指定位置插入新行。

下面的代码演示了 execute()方法的另外一个实例，它从 Employees 表中删除一行。DELETE 语句中的 FROM 子句指明了要删除行的表。

```
var SQLString = "DELETE FROM Employees
      WHERE Last_Name = 'Miller'";
database.execute(SQLString);
```

前面代码中的 SQL 字符串使用 WHERE 子句在表中查找 Last_Name 字段等于“ Miller ”的行。注意前面的语句将真正地从表中删除所有 Last_Name 字段等于“ Miller ”的行。在使用 DELETE 语句时，请在执行它之前确保清楚了真正所要删除的记录。还请在使用 DELETE 语句时确保包括了 WHERE 子句，否则将删除指定表中的所有记录。

提示：尽管 execute()方法不返回查询结果，但是它返回表明了查询操作结果的错误码。请参阅 Netscape 的 LiveWire 文档来了解 execute()方法所返回的一系列错误码。

下一步，将向 GetStudentIDLiveWire.html 文件中添加代码，使用 execute()方法向数据库中写入记录。代码将向数据库中的 Students 表中追加一条学生记录。将使用 Client 对象的 studentID 属性，以及 Request 对象的表单属性来创建每条记录。Request 对象的表单属性

的值来源于 RegistrationLiveWire.html 文件，它调用了 GetStudentIDLiveWire.html 文件。

向 GetStudentIDLiveWire.html 文件中添加代码，使用 execute()方法向数据库中写入记录：

1．返回到文本编辑器或 HTML 编辑器窗口中的 GetStudentIDLiveWire.html 文件。

2．在结束的</SERVER>标签前面，添加下面的语句来创建 SQL 字符串和运行 execute()方法。该语句包含在 else 结构中，当且仅当前面的用来判断数据库是否连接的 if 语句返回 false 值时才执行它。

```
else {
        var SQLString = "INSERT INTO Students VALUES('"
                + client.studentID + "', '"
                + request.last_name + "', '"
                + request.first_name + "', '"
                + request.address + "', '"
                + request.city + "', '"
                + request.state + "','"
                + request.zip + "', '"
                + request.email + "')";
        database.execute(SQLString);
database.disconnect()
client.sameSession="true";
```

3．在结束的</SERVER>标签前，添加下面的代码，它向客户返回一条包含了最近创建的学生 ID 的响应。

```
write("<H2>WebAdventure Computer Training
Registration</H2>");
write("Thanks " + request.first_name + "! Your new
student ID is <B>" + client.studentID + "</B>");
write(". Click <A HREF='CourseListingLiveWire.html'>here
</A> to proceed to the course registration page.
```

4．最后，添加下面的代码结束 else 结构并对 Project 解锁。

```
        }
project.unlock();
```

5．保存并关闭 GetStudentIDLiveWire.html 文件。

下一步，编译注册程序：

1．切换至系统命令提示。不同的操作系统访问命令提示符的方式不同。如果在 Windows

95/98 平台访问命令提示符，请在开始菜单选中运行并输入“ command ”。如果在 Windows NT 访问命令提示符，请在开始菜单选中运行并输入“ cmd ”。

2．切换到 Data Disk 的 Tutorial.11 文件夹。

3．使用 jsac 命令将注册程序编译为 Registration.web。在重新编译该程序的时候，请使用下面的命令，确保包括了 RegistrationLiveWire.html、CourseListingLiveWire.html 和 GetStudentIDLiveWire.html 文件。

```
c:\Netscape \SuiteSpot \bin \https \jsac -v -o
Registration.web RegistrationLiveWire.html
CourseListingLiveWire.html GetStudentIDLiveWire.html
```

帮助：由于空间的限制，前面的代码包括了一个换行。请确保在按下回车键之前在单行上输入整条命令。

4．在程序编译完之后，关闭命令提示窗口。

下一步，安装和运行注册程序：

1．启动应用程序管理器，接着单击顶帧内的“ Add Application ”按钮。

2．在 Add Application 帧内，在姓名框内输入 Registration。接着在 Web File Path 框内，输入 a:\Tutorial.11\Registration.web。

3．在 Default Page 框内输入\RegistrationLiveWire.html。

4．保持 Add Application 表单内的其余编辑框的默认值并单击“ OK ”按钮。

5．在左帧内的已安装应用程序列表中选中 Regsitration 并单击“ Run ”。在 Web 浏览器中打开 StartPage.html。填充注册信息并单击“ Get Student ID ”按钮。将看到与图 11-10 类似的响应。

图 11-10　GetStudentIDLiveWire.html 返回的响应

6．关闭 Web 浏览器窗口。

下一步，将创建 RegisterStudentLiveWire.html 文件。RegisterStudentLiveWire.html 将注册信息追加到 registration 数据库中的 Registration 表中，接着向学生返回一条响应。

提示：在实际的应用环境中，需要包括检测代码来确保学生不对同一课程注册两次。为了简化，即使学生已注册某个课程，RegisterStudentLiveWire.html 中的代码也往 Registration 表中追加新记录。

创建 RegisterStudentLiveWire.html 文件：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．添加起始<SERVER>标签和连接 DSN 文件的语句。

```
<SERVER>
project.lock();
database.connect("ODBC", "WebAdventureLiveWire", "", "",
  "");
if (!database.connected())
        write("The database is not available.");
```

3．下一步，添加下面的代码使用 execute()方法执行 SQL 语句。这些语句将一个含有 SQL 语句的文本字符串赋给 SQLString 变量。SQL 语句使用 INSERT 语句和 INTO 子句来追加 Client 对象的 studentID 属性和包含在 Request 对象的表单属性中的值。Request 对象的表单属性包含了在 CourseListingLiveWire.html 文件内的注册表单域内输入的值。

```
else {
        var SQLString = "INSERT INTO Registration VALUES('"
                + client.studentID + "', '"
                + request.course + "', '"
                + request.days + "', '"
                + request.time + "')";
        database.execute(SQLString);
database.disconnect();
}
```

4．输入下面的向用户返回响应的语句。

```
write("<H2>WebAdventure Computer Training
Registration</H2>");
write("You are registered for " + request.course +
" on " + request.days + ", " + request.time);
write(". To register for another course, click <A
```

```
HREF='CourseListingLiveWire.html'>here</A>. Or click <A
HREF='ReviewScheduleLiveWire.html'>here</A> to review
your current schedule.");
```

5．断开数据库连接，对 Project 解锁并输入结束的<SERVER>标签。

```
project.unlock();
</SERVER>
```

6．在 Data Disk 的 Tutorial.11 文件夹内将文件存为 RegisterStudentLiveWire.html。

7．使用 jsac 命令重新编译注册程序。在执行 jsac 时请确保包括了所有的 4 个文件，如下：

```
c:\Netscape\SuiteSpot\bin\https\jsac -v -o
Registration.web RegistrationLiveWire.html
CourseListingLiveWire.html GetStudentIDLiveWire.html
RegisterStudentLiveWire.html
```

8．使用应用程序管理器重新启动和运行应用程序。在 RegistrationLiveWire.html 页面上，输入前面创建的 studentID 并单击“Class Registration”按钮打开 CourseListingLiveWire.html。图 11-11 显示了在 Navigator 中文档显示的结果。

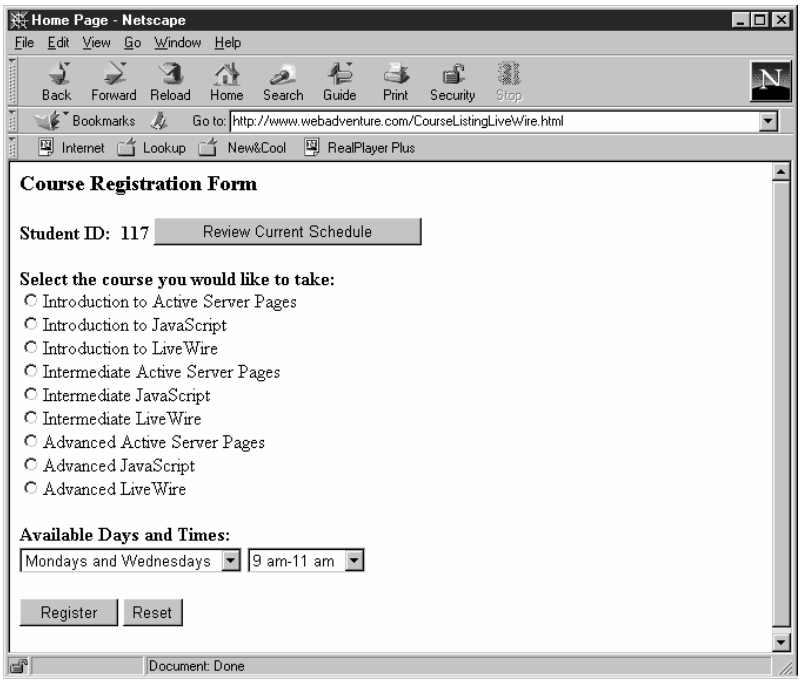


图 11-11　Navigator 中的 CourseListingLiveWire.html

9．填充课程注册表单并单击“ Register ”按钮。图 11-12 显示了一个 Register-StudentLiveWire.html 返回的响应实例。

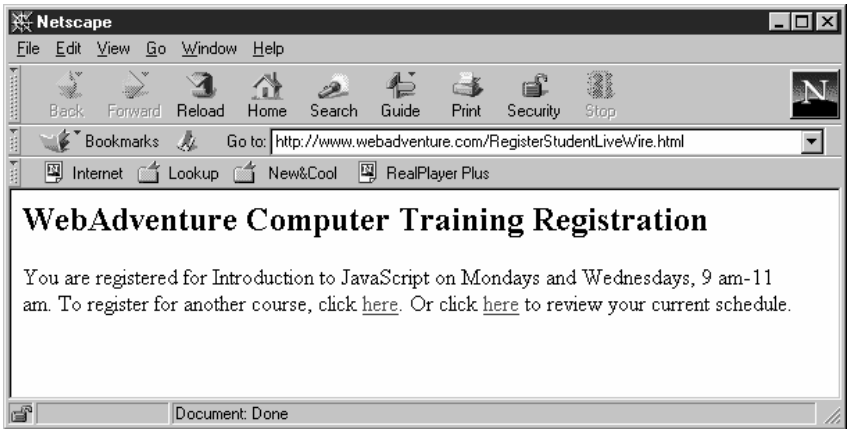


图 11-12　RegisterStudentLiveWire.html 返回的响应

10．关闭 Web 浏览器窗口。

Cursor 方法

execute()方法允许添加、删除和修改表中的记录，但是除了错误码，它不返回任何结果。除了添加、删除或修改记录，还想浏览数据库中的记录并在需要时作出修改或检索值，它们不能够通过 execute()方法实现。为了实现上述类型的功能，必须使用 Database 对象的 cursor()方法来创建 Cursor 对象。Cursor 对象包含了使用 cursor()方法执行的 SELECT 查询所返回的记录。能够浏览 Cursor 对象和检索值或者是添加、删除和修改记录。对 Cursor 对象中的记录集作出的任何修改都将被写入服务器上实际的数据库中。

提示：Cursor 对象中的记录集有时称为虚拟表。

使用 variable = database.cursor("SQL statement", updatable);来创建 Cursor 对象。Cursor 对象被赋给指定的变量名。SQL 语句必须是有效的、用 ODBC SQL 形式编写的 SELECT 语句。updatable 参数是一个布尔值，它指明 Cursor 对象中的记录是否能够被修改。true 值表明 Cursor 对象记录能够被修改；false 值表明 Cursor 对象是只读的。Cursor 对象包括了几种用来对记录集进行操作的方法，如表 11-3 所列出的。

表 11-3　LiveWire Cursor 对象方法

方　法	描　述
close()	关闭 Cursor 对象
columnName(*position*)	返回由位置参数指明的列名
columns()	返回记录集中的列数
deleteRow()	从记录集中删除一行

续表

方　法	描　述
insertRow()	往记录集中插入一行
next()	移动到记录集中的下一条记录
updateRow()	保存对记录集中当前行的修改

应该始终使用的 Cursor 对象方法是 close()方法。在所有 Cursor 对象被关闭之前，不能使用 Database 对象的 disconnect()方法断开数据库连接。例如，下面的语句创建了一个只读的、包含了 Employees 表中记录的 Cursor 对象，然后使用 close()方法关闭 Cursor 对象：

```
employeesTable = database.cursor("SELECT * FROM Employees
    ORDER BY Last_Name, First_Name", false);
statements ;
employeesTable.close();
```

浏览 Cursor　　Cursor 对象中的记录集的位置被称为游标。在使用 cursor()方法首次创建 Cursor 对象时，游标最初放在记录集中的第一行之前。图 11-13 显示了首次在 Cursor 对象中打开 Employees 表时游标放置的位置。

图 11-13　Cursor 对象中最初的游标位置

提示：实际上不可能“看到”如图 11-13 中所示 Cursor 对象中的记录集。图 11-13 中的图解仅用于演示目的。

为了浏览 Cursor 对象中的记录，请使用 next()方法。首次使用 next()方法时，它将游标放在记录集的第一行上。例如，下面的代码创建了一个新的 Cursor 对象，然后将游标移到结果记录集中的第一条记录上。

```
employeesTable = database.cursor("SELECT *
    FROM Employees WHERE Last_Name = 'Miller'", false);
employeesTable.next();
statements ;
employeesTable.close();
```

在使用记录集和 next()方法时，不能确定在游标的当前位置后是否存在另一条记录，

也不能确定 SQL SELECT 语句是否返回记录。例如，前面的代码假定在 Employees 表中存在 Last_Name 字段值为“ Miller ”的记录。若其他人已经删除了该记录，那么 Cursor 对象将不包含任何记录。为了确定存在下一条可访问的记录，若 next()方法在记录集中找到了下一行，那么它返回 true 值，或者它在记录集中未找到下一行那么它返回 false 值。下面的代码演示了在移动游标之前，怎样使用 if 语句来检测 next()方法返回的值。请注意 next()方法是作为 if 语句的条件表达式执行的。还请注意条件表达式未包含比较运算符。因为执行 next()方法时，它自动返回 true 或 false 值，所以不需要比较运算符。

```
employeesTable = database.cursor("SELECT * FROM Employees
        WHERE Last_Name = 'Miller'", false);
if (employeesTable.next()) {
        statements ;
}
employeesTable.close();
```

数据库表中的字段名被赋给了已初始化的 Cursor 对象的属性。例如，若用 Employees 数据库初始化 Cursor 对象 employeesTable，那么接着就能够使用类似于 employeesTable.First_Name 语句来引用 First_Name 字段。请明白任何时候使用 next()方法时，Cursor 对象属性内的每个字段的内容都将变化以便反映当前游标位置上记录的内容。下面的代码演示了一段简单的程序，它将 Employees 表中每个程序员的姓名以及每个程序员所居住的城市返回给客户。程序使用 while 语句来遍历该表。图 11-14 显示了返回给客户的结果。

```
employeesTable = database.cursor("SELECT *
FROM Employees", false);
while(employeesTable.next()) {
        write(employeesTable.First_Name + " "
                + employeesTable.Last_Name + " lives in "
                + employeesTable.City + ", "
                + employeesTable.State + "<BR>");
}
employeesTable.close();
```

下一步，将创建 ReviewScheduleLiveWire.html 文件，它将显示学生注册的课程。

创建 ReviewScheduleLiveWire.html 文件：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．添加起始<SERVER>标签和连接 DSN 文件的语句。

```
<SERVER>
project.lock();
database.connect("ODBC", "WebAdventureLiveWire", "", "", "");
```

```
if (!database.connected())
    write("The database is not available.");
```

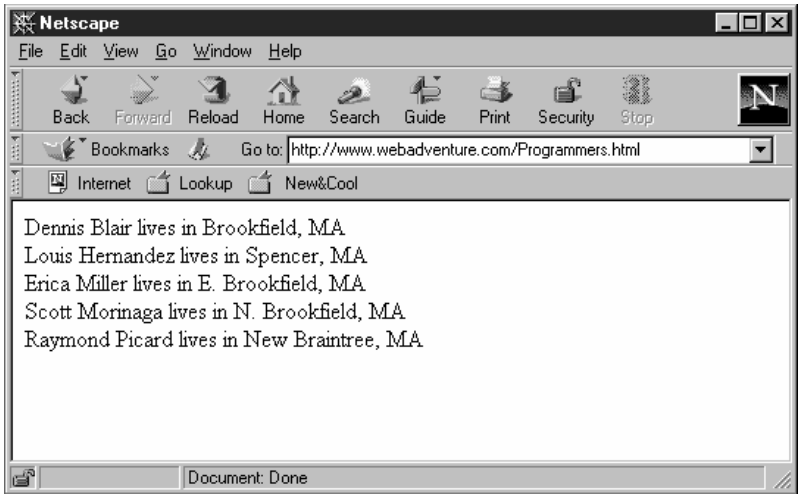


图 11-14 Cursor 对象程序结果

3．添加起始的 else {语句来包括数据库语句。

4．输入 var rsSchedule = database.cursor("SELECT * FROM Registration WHERE Student_ID = '"+ client.studentID + "'", false);来初始化 Cursor 对象。请确保在 Student_ID =子句后输入了一个单引号和一个双引号。还请确保在该语句的 client.studentID 部分后输入了一个双引号，接着是一个单引号，紧接着是一个双引号。

5．添加下面部分，它将响应返回给学生。

```
write("<H2>This is your current schedule</H2>");
while(rsSchedule.next()) {
    write(rsSchedule.Course + ", "
    + rsSchedule.Days + ", "
    + rsSchedule.Time + "<BR>");
}
```

6．关闭 Cursor 对象，断开数据库连接，为 else{语句添加反括弧，对 Project 解锁并结束<SERVER>标签。

```
    rsSchedule.close();
    database.disconnect();
}
project.unlock();
</SERVER>
```

7．在 Data Disk 的 Tutorial.11 文件夹内将文件存为 ReviewScheduleLiveWire.html。

8．使用 jsac 命令重新编译注册程序。在执行 jsac 时，请确保包括了所有的 5 个文件，如下：

```
c:\Netscape\SuiteSpot\bin\https\jsac -v -o
Registration.web RegistrationLiveWire.html
CourseListingLiveWire.html GetStudentIDLiveWire.html
RegisterStudentLiveWire.html ReviewScheduleLiveWire.html
```

9．使用应用程序管理器重新启动和运行应用程序。

10．在 RegistrationLiveWire.html 页面内输入已有的学生 ID 并单击“ Class Registration ”按钮。在 CourseListingLiveWire.html 文档内，单击“ ReviewCurrentSchedule ”按钮。图 11-15 显示了在学生注册了数个班级之后的输出结果。

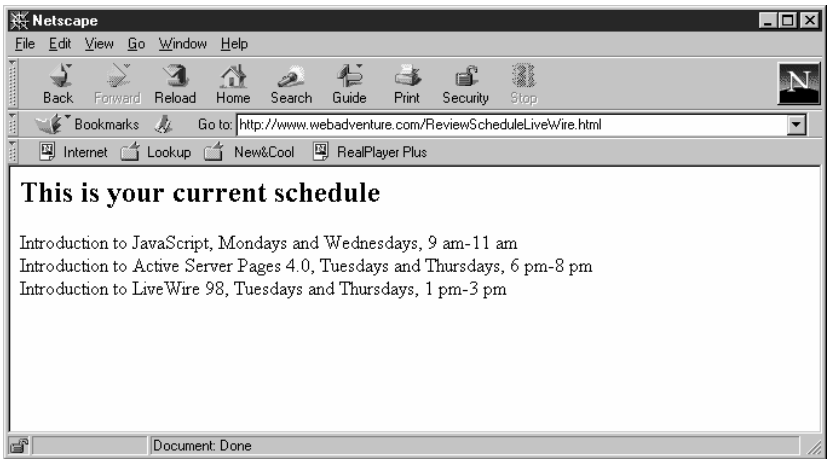


图 11-15　Navigator 中的 ReviewScheduleLiveWire.html

11．关闭 Web 浏览器窗口。

可更新游标　在使用 true 值作为 cursor()方法的 updatable 参数时，将创建一个可更新游标。可更新游标是一个能够被修改的 Cursor 对象记录集。在创建一个可更新游标时，不能在 SELECT 语句内指定关系数据库中的多个表。例外，必须包括键值（如主键）并且 SELECT 语句不能含有用来对查询返回的结果进行排序的 GROUP BY 子句。可更新游标具有三个特有的 Cursor 对象方法：updateRow()、insertRow()和 deleteRow()方法。这三种方法都能够接受一个包含了数据库表名（其中能够更新、插入或删除行）的字符串参数。确保是数据库表名而不是赋给 Cursor 对象的变量名。

updateRow()方法用来保存对记录集内行的任何修改。在修改了字段值之后请执行 updateRow()。例如，假设 Erica Miller 刚结婚并且将她的姓改为“ Lee ”。下面的代码使用了用来检索 Erica Miller 记录的 SELECT 语句和 WHERE 子句创建了一个 Cursor 对象，接着使用 next()方法移到记录集内的第一条记录并更新 Last_Name 字段。

```
employeesTable = database.cursor("SELECT * FROM Employees
        WHERE Last_Name = 'Miller'", true);
if (employeesTable.next()) {
        if (employeesTable.Last_Name == "Miller") {
                employeesTable.Last_Name = "Lee";
                employeesTable.updateRow("Employees");
}
employeesTable.close();
```

如果不是在表尾追加行，而是在记录集内的游标位置处插入新行，insertRow()方法与 SQL INSERT 语句类似。在执行 insertRow()方法之前，将值赋给 Cursor 对象内代表字段的属性。若不为新记录的字段赋给新值，那么将使用字段的当前值。例如，Raymond Picard 的妻子 Lisa 参加了 WebAdventure 编程小组。下面的代码使用 while 循环查找 Raymond Picard 的记录，接着仅修改 Employee_ID、First_Name 和 Extension 字段的值。因为 Raymond 和 Lisa 住在一起，所以不需要修改其余的字段值（地址信息）。

```
employeesTable = database.cursor("SELECT *
FROM Employees", true);
while(employeesTable.Last_Name != "Picard") {
        employeesTable.next();
}
employeesTable.Employee_ID = "107";
employeesTable.First_Name = "Lisa";
employeesTable.Extension = "x309";
employeesTable.insertRow("Employees");
employeesTable.close();
```

提示：若在 Cursor 对象内首次执行 next()方法之前执行 insertRow()方法，那么新行将插在表的开始处，并且未包含值的字段都将赋给空值。

deleteRow()方法与 SQL DELETE 语句类似，除了不是删除所有与 SELECT 语句的结果相匹配的行，deleteRow()仅删除游标所处位置处的记录。例如，若 Scott Morinaga 离开了 WebAdventure，可以使用下面的语句来删除他的雇员记录：

```
employeesTable = database.cursor("SELECT *
        FROM Employees", true);
while(employeesTable.Last_Name != "Morinaga") {
        employeesTable.next();
}
employeesTable.deleteRow("Employees");
employeesTable.close();
```

提示：与 execute()方法一样，updateRow()、insertRow()和 deleteRow()方法返回指示查询操作结果的错误码。请参阅 Netscape 的 LiveWire 文档来了解这些方法所返回的错误码。

SQLTable()方法

SQLTable()方法将 SELECT 语句的结果作为 HTML 表返回给客户。SQLTable()方法是将数据库信息返回给客户的最快捷的途径。HTML 表的行和列分别对应于数据库表的记录和字段，并且包含了一个行标题，其中包含了表中每个字段的名称或标题。请注意不能控制返回给客户的表的格式。下面的代码根据 Employees 数据库表返回一个 HTML 表。图 11-16 显示了返回给客户的结果。

```
write("<H2>WebAdventure Programmers</H2>");
database.SQLTable("SELECT * FROM Employees
        ORDER BY Last_Name, First_Name");
```

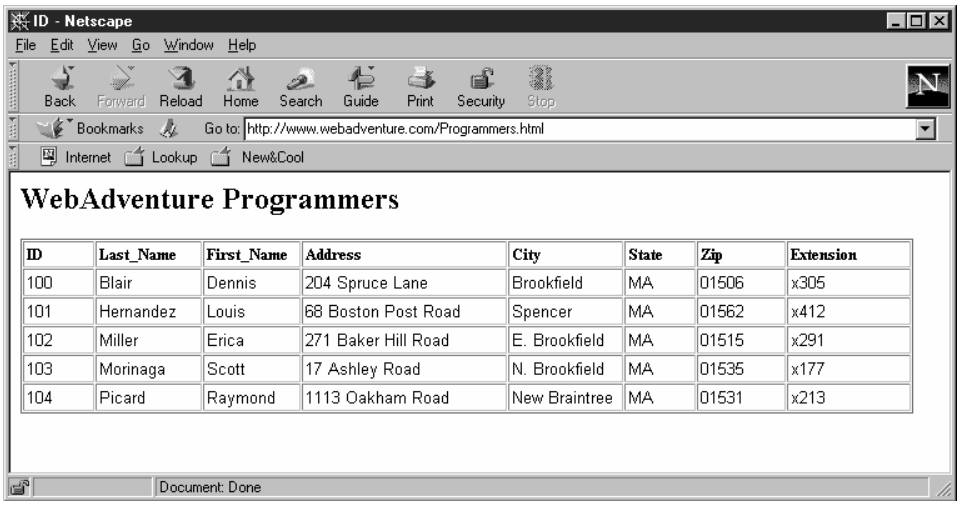


图 11-16　SQLTable()方法的结果

下一步，将修改 ReviewScheduleLiveWire.html 文件以便使用 SQLTable()方法返回学生的课程表：

1．切换到 HTML 编辑器或文本编辑器中的 ReviewScheduleLiveWire.html 文件。

2．删除下面的语句：

```
var rsSchedule = database.cursor("SELECT *
Registration WHERE Student_ID = '"+client.studentID + "'",
        FROM false);
write("<H2>This is your current schedule</H2>");
while(rsSchedule.next()) {
        write(rsSchedule.Course + ", "
        + rsSchedule.Days + ", "
```

```
        + rsSchedule.Time + "<BR>");
}
rsSchedule.close();
```

3．在 database.disconnect()语句前添加下面的语句：

```
write("<H2>This is your current schedule</H2>");
database.SQLTable("SELECT Course, Days, Time FROM
Registration WHERE Student_ID = '" + client.studentID
+ "'");
```

4．在 Data Disk 的 Tutorial.11 文件夹内将文件存为 ReviewScheduleLiveWire.html。

5．重新编译应用程序，接着使用应用程序管理器重新启动运行它。

6．在 RegistrationLiveWire.html 页面内输入一个现有的学生 ID 并单击“ Class Registration ”按钮。在 CourseListingLiveWire.html 文档内单击“ Review Shedule ”按钮。图 11-17 显示了 SQLTable()方法的输出结果。

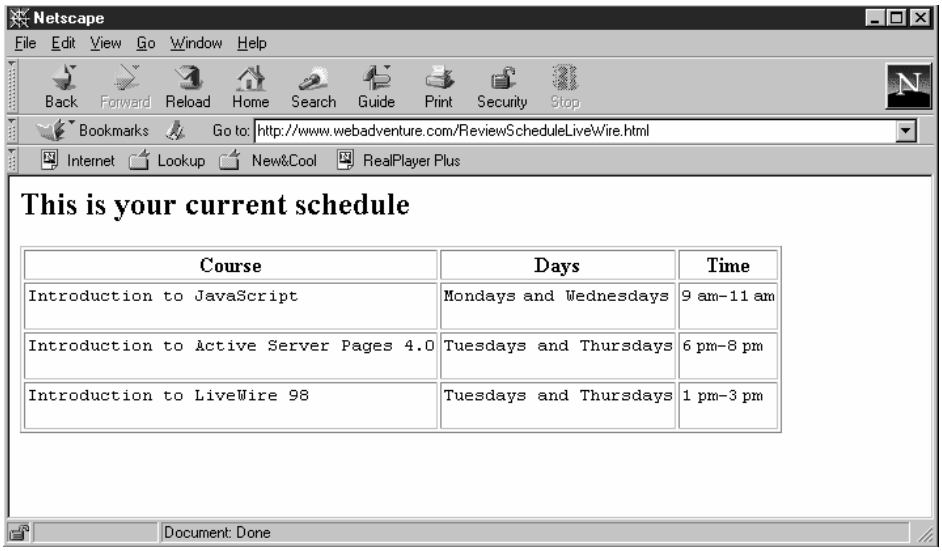


图 11-17　使用 SQLTable()方法的 RegistrationLiveWire.html

7．关闭 Web 浏览器。

11.1.6　LiveWire 事务处理

在使用可更新游标时，默认情况下每条语句随着 LiveWire 的执行被提交给数据库。例如，employeesTable.deleteRow("Employees");语句将立即和永久性地删除数据库中游标当前位置处的行。在永久性地修改数据库记录时，请记住是提交修改。大多数数据库具有众所

周知的自动提交特性，即立即提交对数据库记录的修改，或是若存在错误就取消修改。可以使用事务来替代数据库的默认自动提交特性。事务将数据库语句作为组提交。若正在保存到数据库中的数据存在某种错误，还可以取消或返回事务。没有事务，若发生问题，就无法撤销对数据库的修改。任何时候添加、删除或修改数据库中的记录都使用事务是一个不错的主意。事务保证所有相关联的数据库修改一起成功或是一起失败。

使用 beginTransaction()方法来开始一个事务。跟在 beginTransaction()方法后的所有数据库语句被称为当前事务。在作为事务部分的数据库语句后，执行 commitTransaction()方法来提交对数据库的修改。在 beginTransaction()方法后的任何地方，可以执行 rollBackTransaction()方法来取消作为当前事务部分的所有修改。例如，若当前事务漏掉了所必需的字段，如键字段，那么接着可以调用 rollBackTransaction()来取消作为当前事务部分的所有修改。请注意若试图在使用 beginTransaction()之前调用 commitTransaction()或 rollBackTransaction()，那么将会收到来自数据库的错误。还请注意事务的范围局限于当前请求。一旦响应返回给客户，那么就再不能返回一个事务。另外，若响应在执行 commitTransaction()方法或 rollbackTransaction()方法之前就返回给客户，那么事务将自动被提交给数据库（只要用来初始化可更新游标的 cursor()方法的 updatable 被设置为 true）。

提示：即使不使用 LiveWire 的事务处理方法，Informix ANSI 数据库也自动使用事务。

下面的代码演示了怎样使用事务处理来向 Employees 表中添加新雇员记录：

```
employeesTable = database.cursor("SELECT *
FROM Employees", true);
database.beginTransaction();
while(employeesTable.Last_Name != "Picard") {
        employeesTable.next();
}
employeesTable.Employee_ID = "107";
employeesTable.First_Name = "Lisa";
employeesTable.Extension = "x309";
employeesTable.insertRow("Employees");
database.commitTransaction();
employeesTable.close();
```

前面的代码未使用 rollbackTransaction()方法。在下节将学习怎样返回事务，它与错误处理相关。

11.1.7　LiveWire 错误处理

Database 对象的许多方法，包括 insertRow()、deleteRow()和 updateRow()方法都以状态

码的形式返回错误消息。返回值 0 表示操作成功完成。任何其他状态码都表示一个错误。表 11-4 列出了 LiveWire 状态码和它们的描述。

表 11-4 LiveWire 状态码

状 态 码	描 述
0	无错
1	内存不足
2	未初始化对象
3	类型转换错误
4	未注册数据库
5	服务器报出的错误
6	来自服务器的消息
7	来自厂商库的错误
8	丢失连接
9	读取结束
10	无效的对象使用
11	列不存在
12	对象内的无效定位（绑定错误）
13	不支持的特性
14	无效的引用参数
15	未发现数据库对象
16	所需信息丢失
17	对象不支持多个访问
18	对象不支持删除
19	对象不支持插入
20, 21	对象不支持更新
22	对象不支持索引
23	对象不能被停止
24	所提供的连接不正确
25	对象不支持权限
26	对象不支持游标
27	不能打开

在返回非 0 的状态码时，可以使用 majorErrorCode()、majorErrorMessage()、minorErrorCode()和 minorErrorMessage()方法来获取更多的、有关错误的信息。majorErrorCode 和 majorErrorMessage()方法分别返回来自服务器的最近一个主要的错误代码和消息。任何次要的错误码和错误消息都能够通过 minorErrorCode()和 minorErrorMessage()方法返回。例如，为了将主要的错误消息返回给客户请使用与 write(majorErrorMessage())；类似的语句。

提示：不要混淆 LiveWire 状态码和 majorErrorCode()和 minorErrorCode()所返回的值。状态码是 LiveWire 所返回的错误码。而 majorErrorCode()和 minorErrorCode()所返回的值是数据库服务器所返回的。

在执行事务时若发生问题，可以使用状态码和错误方法以及 rollbackTransaction()方法来取消对数据库的修改。下面的代码显示了一个使用 rollbackTransaction()方法的事务实例。代码将 insertRow()方法返回的状态码赋给 returnedStatus 变量。若 returnedStatus 等于 0，那么调用 commitTransaction()方法并返回一条消息给客户，表明事务执行成功。然而，若 returnedStatus 不等于 0，那么将给客户返回一条描述事务失败的消息以及主要的错误码和消息。

```
employeesTable = database.cursor("SELECT *
FROM Employees", true);
database.beginTransaction();
while(employeesTable.Last_Name != "Picard") {
   employeesTable.next();
}
employeesTable.Employee_ID = "107";
employeesTable.First_Name = "Lisa";
employeesTable.Extension = "x309";
employeesTable.insertRow("Employees");
if (returnedStatus == 0) {
      database.commitTransaction();
        write("The record was added successfully.");
}
else {
      database.rollbackTransaction();
      write("The record was not successfully added.
The database server returned the following
error code and message:<BR>"
            + "error code: " + database.majorErrorCode()
+ "<BR>" + "error message: "
+ database.majorErrorMessage());
}
employeesTable.close();
```

11.1.8　总结

- ✧　数据库是一个有序的信息集合，计算机程序能够迅速地从中访问信息。
- ✧　数据库中的一条记录包含了一个完整的相关信息集。

- ✧ 存储在记录中独立的信息片断称为字段。
- ✧ 平面数据库在单个表中存储信息。
- ✧ 关系数据库跨多个相关的表存储信息。
- ✧ 对主表的每条记录而言，当相关表内的每条记录仅包含一条记录时，那么在两个表之间存在一对一关系。
- ✧ 当主表中的一条记录与相关表中的多条记录有联系时，那么在两个表之间存在一对多关系。
- ✧ 为了减少冗余和重复信息而将表分割为多个相关表称为规范化。
- ✧ 在一个表中的多条记录与另一个表内的多条记录相关时，那么在两个表之间存在多对多关系。
- ✧ 连接表包含了多对多关系中两个表的一对多关系。
- ✧ 用来创建、访问和管理数据库的应用程序或应用程序集称为数据库管理系统（DBMS）。
- ✧ 查询是结构化的命令集和用来检索、添加、修改和删除数据库信息的规则。
- ✧ 报表是数据库表或查询结果的格式化打印输出。
- ✧ 大多数数据库管理系统使用数据操纵语言（DML）来创建查询。
- ✧ 结构化查询语言（SQL，发音与单词“ sequel ”相似）已经成为众多数据库管理系统之间的标准数据操纵语言。
- ✧ 开放数据库连接（ODBC）允许所编写的遵守标准的应用程序访问任何数据源，与之对应存在一个 ODBC 驱动程序。
- ✧ 数据源名（DSN）包含了 Windows 操作系统用来访问特殊的、与 ODBC 兼容的数据库的配置信息。
- ✧ DbPool 和 Connection 对象被用来创建和管理数据库连接池。Database 对象被用来创建和管理客户和数据库之间的每个连接。
- ✧ 标准连接允许多个用户同时访问数据库。串行连接在任何时刻仅允许单个用户访问数据库。
- ✧ execute()方法向数据库管理系统发送用于处理的 SQL 语句。
- ✧ 通过 execute()方法执行的语句被称为转移 SQL，因为它们并不直接被传递给数据库管理系统。
- ✧ Cursor 对象包含了通过 cursor()方法执行的 SELECT 查询所返回的记录。
- ✧ Cursor 对象的记录集的位置被称为游标。
- ✧ 可更新游标是一个能够被修改的 Cursor 对象记录集。
- ✧ SQLTable()方法以 HTML 表将 SELECT 语句的结果返回给客户。
- ✧ 事务以组提交数据库语句。
- ✧ 在 beginTransaction()方法后的所有数据库语句被称为当前事务。
- ✧ Database 对象的许多方法都以状态码的形式返回错误消息。返回值 0 表示操作成功地执行。任何非 0 的状态码指示一个错误。

✧　在返回非 0 的状态码时，可以使用 majorErrorCode()、majorErrorMessage()、minorErrorCode()和 minorErrorMessage()方法来获取更多的有关错误的信息。

11.1.9　问题

1．存储在数据库记录内的独立的信息片断的正确术语是什么？

a．元素

b．字段

c．段

d．容器

2．平面文件数据库由多少个表组成？

a．1

b．2

c．任意个表

d．平面文件数据库不是由表组成

3．当一个表的主键存储在另一个表时，它的名称是什么？

a．键符号

b．记录链接

c．外键

d．惟一标识符

4．为了减少冗余和重复信息而将表分割为多个相关表被称为____。

a．规范化

b．冗余设计

c．分离

d．简化

5．对主表的每条记录而言，当相关表内的每条记录仅包含一条记录时，那么在两个表之间存在____关系。

a．一对无

b．一对一

c．一对多

d．多对多

6．当主表中的一条记录与相关表中的多条记录有联系时，那么在两个表之间存在____关系。

a．一对无

b．一对一

c．一对多

d．多对多

7．当一个表中的多条记录与另一个表内的多条记录相关时，那么在两个表之间存在____关系。

a．一对无

b．一对一

c．一对多

d．多对多

8．____包含了多对多关系中两个表的一对多关系。

a．联合数据库

b．平面文件链接

c．连接表

d．桥表

9．用来创建、访问和管理数据库的应用程序或应用程序集合被称为____。

a．shell 程序

b．大型机系统

c．数据库管理系统

d．三层客户服务器设计

10．大多数数据库管理系统采用____形式的数据操纵语言。

a．CGI 脚本

b．C/C++语法

c．结构化查询语言

d．Java 编程

11．___包含了 Windows 系统用来访问特殊的、与 ODBC 兼容的数据库的配置信息。

a．动态链接库

b．SQL 容器

c．ODBC 接口单元

d．数据源名

12．在 LiveWire 中连接数据库的正确语法是什么？

a．database.connect(*arguments*);

b．Database.connect(*arguments*);

c．database.Connect(*arguments*);

d．Database.Connect(*arguments*);

13．当连接至 Windows 平台上 ODBC 数据库时，服务器名参数应设为什么？

a．数据库内的表名

b．Windows NT 服务器名

c．源数据库文件名

d．DSN 名

14．若成功连接了数据库，那么哪种方法将返回 true 值？

a．connect();
b．connected();
c．success();
d．dbAvailable();
15．____连接允许多个用户同时访问数据库。
a．多
b．串行
c．标准
d．并行
16．____连接任何时刻仅允许一个用户访问数据库。
a．多
b．串行
c．标准
d．并行
17．通过 execute()方法执行的语句被称为____SQL 语句。
a．转移
b．直接
c．数据库
d．服务器端
18．若对 SQL DELETE 语句不使用 WHERE 字句，那么将会发生什么情况？
a．将收到错误
b．目标表内的所有行被删除
c．目标表内没有行被删除
d．什么也不发生
19．____对象包含了通过 cursor()方法执行的 SELECT 查询作为记录集所返回结果。
a．游标
b．记录
c．表
d．数据库
20．哪种方法被用来浏览记录集？
a．goto()
b．moveTo()
c．record()
d．next()
21．怎样创建可更新游标？
a．使用 curosr()方法
b．使用 true 值作为 curosr()方法的 updatable 参数

c．在 cursor()方法内包括“ update ”字符串参数
d．不能创建可更新游标
22．下面的哪种方法不是可更新游标特有的？
a．updateRow()
b．insertRow()
c．deleteRow()
d．next()
23．哪种方法将查询结果作为 HTML 表返回？
a．HTMLTable()
b．SQLTable()
c．SQLtoTable()
d．ReturnTable()
24．哪种方法取消事务？
a．cancel()
b．discard()
c．rollback()
d．rollbackTransaction()
25．哪种错误码表示成功完成数据库操作？
a．OK
b．SUCCESS
c．1
d．0

11.1.10 练习

1．在本教程中注册程序内所创建的新学生 ID 是根据 Project 对象的 idNum 属性生成。根据 Project 对象的属性生成新学生 ID 并不是创建它最好的方法，因为每次重新启动应用程序都将重新初始化 idNum 的值。修改注册程序以便根据数据库中的 Students 表来创建新学生 ID 而不是根据 Project 对象的 idNum 属性。将需要使用 next()方法遍历 Students 表内的记录以便检索最后一个赋给学生的 ID。对此练习，请使用 11.1 节中所使用的 WebAdventureLiveWire DSN。

2．数据库设计技术包括了识别和设计五种规范化级别的过程：第一范式、第二范式、第三范式、第四范式和第五范式。搜索 Internet 或从图书馆查阅与这些技术相关的信息。撰写一篇描述怎样识别和设计每种范式的论文。

3．电子商务是现在 Web 开发中一个热门话题，因为多数企业希望能够在线销售它们的产品。在 Internet 上搜索有关这方面主题信息并撰写一篇论文描述电子商务的现状，包括流行的数据库管理系统、安全问题以及此种技术发展到何种程度。

4．创建将网址客户簿记录保存到数据库的 LiveWire 程序。

5．创建将记录保存到数据库的电话目录应用程序。在数据库中应该包括标准的电话目录字段如姓名、地址、城市、州、邮编和电话号码等。创建一个 HTML 文档作为主“ 目录 ”，其中能够选择和检索记录。请创建一个 HTML 文档，能够使用它往数据库添加记录；以及另外一个 HTML 文档，其中能够编辑记录。请使用任何能够使用的、与 ODBC 兼容的数据库（如 Paradox 或 Access）来创建数据库表。

6．为在线书店创建一个购物车应用程序。以本节所创建的注册程序为模型。然而，不是将课程注册添加到数据库中，而是订购信息。请为不同类型的书籍使用不同的页面。购物车应列出用户想购买的书籍并提供一种将信息写入数据库和给客户返回响应的“ 校验 ”机制。对前面的 4 个练习，请使用能够访问的数据库管理系统（如 Access、Paradox 或 SQL Server）来创建保存数据的数据库文件。

11.2　使用 ASP 连接数据库

本节目标

在本节将学习：

- ✧ 怎样使用 Active Server Pages 连接数据库
- ✧ ADO Connection 对象
- ✧ 怎样通过 ADO 执行 SQL 命令
- ✧ 怎样通过 ADO 执行事务处理
- ✧ 怎样通过 ADO 处理数据库错误

11.2.1　使用 Active Server Pages 连接数据库

Active Server Pages（ASP）使用 ActiveX Data Objects 访问数据库。ActiveX Data Objects（ADO）是 Microsoft 数据库连接技术，它让 ASP 和其他 Web 开发工具能够访问与 ODBC 和 OLE DB 兼容的数据库。OLE DB 是 Microsoft 提出的、作为 ODBC 继任的数据源连接标准。OLE DB 和 ODBC 之间的主要区别之一是 ODBC 仅支持访问关系数据库而 OLE DB 不仅支持访问关系数据库而且还支持非关系数据源，如电子表格应用程序。将 ASP 数据库连接与 LiveWire 数据库连接相比，我们将重点讨论 ADO 怎样访问 ODBC 数据库。注意本节的许多实例与在 11.1 节中所看到的实例相似。使用相同的实例是为了演示怎样使用两种不同的服务器端 JavaScript 来执行相同的功能。

ADO 和 OLE DB 都是 Microsoft 的通用数据访问策略的一部分，它们用来提供对数据的访问，而不管数据的存储格式。组成通用数据访问技术的组件称为 Microsoft 数据访问组

件（MDAC）。MDAC 随无数的产品捆绑，包括 Windows NT 4.0 选项包、Internet Explorer 4.0、Internet Information Server 4.0 和 Microsoft Visual Studio 6.0。这些产品的大部分（包括 Internet Explorer）都自动安装 MDAC。然而，若确定不了是否在系统上安装了 MDAC，那么可以从 Microsoft 的通用数据访问网址 http://www.Microsoft.com/data 下载组件检验器程序。组件检验器帮助确定在系统上已安装的 MDAC 的组件和版本。还能够从 Microsoft 的通用数据访问网址下载最新版本的 MDAC 和查阅更多的、有关 Microsoft 数据访问技术的信息。

在深入讨论 ADO 对象和集合之前，需要创建一个基于文件的 DSN，将在本节所创建的 ASP 版的数据库程序中使用它。从技术上讲，能够使用 LiveWire 和 ASP 两种版本的程序来访问 DSN。然而，为了避免混淆将为 ASP 版的程序创建一个名为 WebAdventureASP.dsn 的 DSN。

为 ASP 数据库程序创建基于文件的 DSN：

1．单击 Windows 的开始菜单，接着选中设置文件夹下的控制面板。

2．单击或双击（取决于是怎样配置桌面的）控制面板窗口中的 ODBC 图标。

3．在 ODBC 数据源管理器窗口中，选中文件 DSN 标签，接着单击“ 添加 ”按钮。在弹出的创建新数据源窗口内，选中 Microsoft Access Driver（*.mdb）并单击“ 下一步 ”按钮。

4．在创建新数据源窗口内，输入 WebAdventureASP 作为新 DSN 文件的名称并单击“ 下一步 ”按钮。接着单击创建新数据源窗口内最后一个界面上的“ 完成 ”按钮。

5．在弹出的 ODBC Microsoft Access 安装界面内，单击“ 选择 ”按钮。在弹出的选择数据库对话框内，选中 Data Disk 的 Tutorial.11 文件夹下的 WebAdventureCourse.mdb 文件并单击“ 确定 ”。

6．单击“ 确定 ”按钮关闭 ODBC Microsoft Access 安装界面。

7．单击“ 确定 ”关闭 ODBC 数据源管理器窗口。接着关闭控制面板。

为了让某些编程语言使用 ADO，MDAC 在系统上安装了几个必需的文件。使用 ASP 访问数据库所必需的 ADO 文件名为 adojavas.inc。默认情况下，ADO 文件被安装在 C:\Program Files\Common Files\System\ado 下。当创建需要访问数据库的 ASP 时，必须将 adojavas.inc 文件拷贝到 ASP 文件所在的目录下并使用#include 命令将文件插入到程序中。#include 命令指定了一个在服务器处理 ASP 文件之前插入到其中的文件。在 ASP 文档内包括一个文件的语法是<!-- #include file ="filename" -->。请在 ASP 文件的处理命令之前，但在脚本界定符之前放置#include 命令。下面的代码演示了怎样使用#include 命令将 adojavas.inc 文件插入到 ASP 文件中准备访问数据库。代码假定 adojavas.inc 文件位于与 ASP 文档相同的服务器目录下。

```
<%@ LANGUAGE=JScript %>
<!--#include file="adojavas.inc"-->
<%
ASP statements;
```

```
%>
```

下一步，将 adojavas.inc 文件拷贝到 Tutorial.11 文件夹内：

1．确定计算机上 adojavas.inc 文件的位置。默认安装目录是 C:\Program Files\Common Files\System\ado。

2．在确定了 adojavas.inc 的位置之后，立即将文件拷贝到 Data Disk 的 Tutorial.11 文件夹内。

下一步，创建主 RegistrationASP.html 文档：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．输入文档的起始<HTML>、<HEAD>和<BODY>段。

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

3．添加下面组成文档的标签、文本和表单。两个表单都使用提交按钮来调用 ASP 文档。在本节的后面部分，将创建提交按钮所调用的 ASP 文档。

```
<H2>WebAdventure Computer Training Registration</H2>
<H3>Welcome</H3>
<P>Welcome to Computer Training at WebAdventure! We offer
a variety of computer training and technology courses
that focus on the Web. To sign up for a course, please
fill out the New Student Registration form and click the
<B>Get Student ID</B> button to obtain a student ID. If
you are a current student, enter your student ID number
and click the Class Registration button to register for
new classes or to review your current schedule.</P>
<H3>New Student Registration</H3>
<FORM NAME="customerInfo" METHOD="post"
ACTION="GetStudentIDASP.asp">
Last Name:  <INPUT TYPE="text" NAME="last_name"
SIZE=30>  
First Name:  <INPUT TYPE="text" NAME="first_name"
SIZE=30><BR>
Address:  <INPUT TYPE="text" NAME="address" SIZE=30>
City, State, Zip:  <INPUT TYPE="text" NAME="city"
SIZE=20>
<INPUT TYPE="text" NAME="state" SIZE=2 MAXLENGTH=2>
```

```
<INPUT TYPE="text" NAME="zip" SIZE=5 MAXLENGTH=5><BR>
E-Mail:  <INPUT TYPE="text" NAME="email" SIZE=50><P>
<INPUT TYPE="submit" NAME="submit" VALUE="
Get Student ID ">
<INPUT TYPE="reset">
</FORM>
<H3>Returning Student Registration</H3>
<FORM METHOD="post" ACTION="CourseListingASP.asp">
Student ID:  <INPUT TYPE="text" NAME="id">
<INPUT TYPE="submit" VALUE=" Class Registration ">
</FORM>
```

4．添加下面的代码结束<BODY>和<HTML>标签：

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

5．在 Data Disk 的 Tutorial.11 文件夹内将文件存为 RegistrationASP.html，接着将文件拷贝或上传到 ASP 服务器。在 Web 浏览器内打开来自服务器的 RegistrationASP.html。图 11-18 显示了文档在 Internet Explorer 中的输出结果。在执行表单按钮之前，需要创建 ASP 文档。

图 11-18　Internet Explorer 中 RegistrationASP.html

6．关闭 Web 浏览器窗口。

ADO 对象模型

ADO 技术基于由访问和维护数据源的对象和集合组成的对象模型。表 11-5 列出了 ADO 对象模型中的对象，表 11-6 列出了集合。在本节将使用其中几个 ADO 对象和集合。

表 11-5　ADO 对象

对　象	描　述
Connection	提供对数据源的访问
Command	执行一个对数据源操作的命令，如 SQL 命令
Parameter	为 Command 对象所执行的命令提供参数
Recordset	根据 SQL 查询结果创建一个可浏览的虚拟表
Field	代表 Recordset 对象的数据库字段
Error	代表从数据库所返回的错误
Property	代表 ADO 对象属性

表 11-6　ADO 集合

集　合	描　述
Errors	数据库所返回的错误
Parameters	与 Parameter 对象相关联的参数
Fields	与 Field 对象相关联的数据库字段
Properties	ADO 对象属性

提示：本教程仅概括地介绍了使用 ASP 和 ADO 访问数据的方法。要获取更多的与使用 ADO 访问 ASP 数据库的相关的内容，请参阅 Microsoft Developer Network http://msdn.microsoft.com/。

在学习怎样使用 ADO 访问数据库之前，需要创建 CourseListingASP.asp 文件，学生使用它来选择他们想要学习的课程。

创建 CourseListingASP.asp 文件：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．输入 ASP 处理命令、起始<HTML>标签和 ASP 段。if 语句使用逻辑非（!）运算符来判断在 Session 对象的 Contents 集合中是否存在 sameSession 变量。若不存在 sameSession 变量，将创建它并将它赋为 true。接着将 Request 对象的 Form 集合的 id 属性赋给 Session 对象的 Contents 集合的 studentID 变量。Request 对象的 Form 集合的 id 变量包含了学生在 Registration 表单中 id 域输入的内容。在用户浏览组成程序的页面时，将在整个注册程序中使用 studentID 变量来跟踪用户。

```
<%@ LANGUAGE=JScript %>
<!-- #include file="adojavas.inc" -->
<HTML>
```

```
<%
if (!Session.Contents("sameSession")){
        Session.Contents("sameSession") = "true";
Session.Contents("studentID ") = parseint(Request.
Form ("id "));
}
%>
```

帮助：ASP 集合变量是以文本数据类型存储的。因此，在将 ASP 集合变量中的值拷贝到在计算中使用的 JavaScript 变量时，必须使用数据类型转换函数。这是 JavaScript 不自动将变量转换为合适的类型少数情况中的一种。前面代码中的 parseInt()函数是一种数据类型转换函数，它将 Request.Form 集合中的 id 变量转换为整数数据类型。请参阅附录 A 中的 JavaScript 数据类型转换函数和方法来了解更多的有关 parseInt()和其他 JavaScript 类型转换函数和方法的内容。

3．添加下面的 HTML 标签、文本和表单。该表单包含了两个用于显示学生课程表的元素。使用输出命令（<%=）将学生的 ID 打印在表单内的屏幕上。studentID 变量被存放在隐藏的表单域内，它将与其他表单域一起被提交给 ReviewScheduleASP.asp 文件。

```
<BODY>
<H3>Course Registration Form</H3>
<FORM METHOD="post" ACTION="ReviewScheduleASP.asp">
<B>Student ID:  <%= Session.Contents("studentID")
%></B>
<INPUT TYPE="hidden" NAME="id" VALUE="<%=
Session.Contents('studentID') %>">
<INPUT TYPE="submit" VALUE=" Review Current Schedule
"><P>
</FORM>
```

4．输入下一个表单，学生使用它来注册课程。表单的内容被提交给 ASP 脚本 RegisterStudentASP.asp。请注意该表单还包含了一个隐藏的、存放 studentID 变量的表单域。

```
<FORM METHOD="post" ACTION="RegisterStudentASP.asp">
<INPUT TYPE="hidden" NAME="id" VALUE="<%=
Session.Contents('studentID') %>">
<B>Select the course you would like to take:</B><BR>
<INPUT TYPE="radio" NAME="course" VALUE="Introduction
to Active Server Pages 4.0">Introduction to
Active Server Pages<BR>
<INPUT TYPE="radio" NAME="course" VALUE="Introduction
to JavaScript">Introduction to JavaScript<BR>
```

```
<INPUT TYPE="radio" NAME="course" VALUE="Introduction to
LiveWire 98">Introduction to LiveWire<BR>
<INPUT TYPE="radio" NAME="course" VALUE="Intermediate
Active Server Pages 4.0">Intermediate Active Server
Pages<BR>
<INPUT TYPE="radio" NAME="course" VALUE="Intermediate
JavaScript">Intermediate JavaScript<BR>
<INPUT TYPE="radio" NAME="course" VALUE="Intermediate
LiveWire 98">Intermediate LiveWire<BR>
<INPUT TYPE="radio" NAME="course" VALUE="Advanced Active
  Server Pages">Advanced Active Server Pages<BR>
<INPUT TYPE="radio" NAME="course" VALUE="Advanced
JavaScript">Advanced JavaScript<BR>
<INPUT TYPE="radio" NAME="course" VALUE="Advanced
LiveWire">Advanced LiveWire<P>
<B>Available Days and Times:</B><BR>
<SELECT NAME="days">
<OPTION SELECTED VALUE="Mondays and Wednesdays">Mondays
and Wednesdays
<OPTION VALUE="Tuesdays and Thursdays">Tuesdays and
Thursdays
<OPTION VALUE="Wednesdays and Fridays">Wednesdays and
Fridays
</SELECT>
<SELECT NAME="time">
<OPTION SELECTED VALUE="9 am-11 am">9 am-11 am
<OPTION VALUE="1 pm-3 pm">1 pm-3 pm
<OPTION VALUE="6 pm-8 pm">6 pm-8 pm
</SELECT><P>
<INPUT TYPE="submit" VALUE=" Register ">
<INPUT TYPE="reset">
</FORM>
```

5．添加下面的代码结束<BODY>和<HTML>标签：

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

6．在 Data Disk 的 Tutorial.11 文件夹内将文件存为 CourseListingASP.asp，接着将文件拷贝或上传到 ASP 服务器。在打开文件之前，需要编写生成新的学生 ID 的 ASP 脚本。

7．关闭文本编辑器和 HTML 编辑器窗口。

11.2.2 ADO Connection 对象

使用 ADO Connection 对象被 ASP 用来访问数据库并且在功能上与 LiveWire 的 Database 对象相似。Connection 对象包含了用来访问和维护数据库的各种方法和属性，如表 11-7 和表 11-8 所示。

表 11-7 Connection 对象方法

方 法	描 述
Open()	打开数据源连接
Close()	关闭数据源连接
Execute()	执行对数据源操作的命令
BeginTrans()	开始一个事务
CommitTrans()	提交一个事务
RollbackTrans()	回滚事务

表 11-8 Connection 对象属性

属 性	描 述
ConnectionString	Open()方法的连接字符串参数
ConnectionTimeout	在放弃连接尝试之前所等待的秒数
CommandTimeout	在放弃命令尝试之前所等待的秒数
State	数据源连接状态：adStateOpen 表示成功的连接，adStateClosed 表示失败的连接
Provider	连接中所使用的数据源
Version	ADO 版本号
CursorLocation	返回表示是客户管理游标、服务器管理游标还是不管理游标的值

在 ASP 中使用数据库的第一步是创建 Connection 对象实例。使用 Server 对象的 CreateObject()方法来创建 Connection 对象实例。在使用 CreateObject()方法时，给它传递一个完整的、系统上组件的程序标识符（progID）。progID 的格式为厂商.组件.版本。连接 ADO 数据库的 progID 是 ADODB.Connection，其中 ADODB 是厂商，Connection 是所要创建的组件（ADODB Connection 对象不需要版本）。使用 CreateObject()方法创建 Connection 对象实例的完整语法是 var object = Server.CreateObject("ADODB.Connection");。使用 object 作为变量名以便在 ASP 代码中引用此连接。

提示：对于所有的 JavaScript 代码，ADODB.Connection progID 是区分大小写的。

在创建了 Connection 对象实例之后，必须使用 Open()方法来打开特定的数据源。Open()方法接收一个包含了连接字符串的字符串参数。连接字符串可以由不同的参数组成（取决

于怎样连接数据库），包括数据库源供给者、数据库文件名、远程供给者以及远程服务器路径。还能够在 Open()方法中包括其他参数，包括用户 ID、口令和其他选项。在使用文件 DSN 时，简单地包括一个 FILEDSN=filename.dsn 连接字符串就行了。在使用完数据库之后，请立即使用 Close()方法来关闭连接。下面语句是一个怎样连接和断开一个基于文件的 Accounting DSN 的实例，其中将连接赋给了 dbConnection 变量。

```
var dbConnection = Server.CreateObject("ADODB.Connection");
dbConnection.Open("FILEDSN=Accounting.dsn");
additional statements ;
dbConnection.Close();
```

与 LiveWire 一样，使用 ASP 一个不错的习惯是确保在试图读取、写入或修改记录之前已成功地连接了数据库。若数据库连接成功，那么 Connection 对象的 State 属性将返回一个 adStateOpen 值；若数据库连接失败，那么返回的将是 adStateClosed 值。下面的代码添加了一个在试图连接数据库之后检测 State 属性的 if 语句。若连接失败，那么 ASP 的 Response.Write()方法将返回一条消息给客户。

```
var dbConnection = Server.CreateObject("ADODB.Connection");
dbConnection.Open("FILEDSN=Accounting.dsn");
if (dbConnection.State == "adStateClosed")
        Response.Write("The database is not available.");
additional statements ;
dbConnection.Close();
```

在 11.1 节，学习了使用 LiveWire 的 lock()和 unlock()方法来决定客户是使用标准还是串行连接来访问数据库。ASP 并不像 LiveWire 那样管理标准和串行连接或具有任何数据库池函数。相反，ASP 依赖 ODBC 驱动程序来处理数据库池和其他连接函数。这意味着不能在 ASP 中包括用来处理多个请求访问同一个数据源的代码。然而，若过去一段指定的时间后还未成功连接数据库，那么可以使用 Connection 对象的 ConnectionTimeout 属性来放弃客户的数据库连接尝试。若网址承受了大量的网络传输或服务，那么将发现使用 ConnectionTimeout 属性是必要的。将一个表示在结束客户连接之前所等待的秒数的整数赋给 ConnectionTimeout 属性。在过去赋给 ConnectionTimeout 属性的秒数之后，将放弃连接尝试并返回一条错误消息给客户。默认情况下，在放弃连接尝试之前 ADO 等待 15 秒。下面的代码演示了前面的 ConnectionTimeout 属性为 30 秒的实例。

```
var dbConnection = Server.CreateObject("ADODB.Connection");
dbConnection.ConnectionTimeout = 30;
dbConnection.Open("FILEDSN=Accounting.dsn");
if (dbConnection.State == "adStateClosed")
        Response.Write("The database is not available.");
```

```
additional statements ;
dbConnection.Close()
```

下一步，开始创建 GetStudentIDASP.asp 文件，它生成新的学生 ID：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．输入 ASP 处理命令和起始的 ASP 界定符：

```
<%@ LANGUAGE=JScript %>
<!-- #include file="adojavas.inc" -->
<%
```

3．添加下面的代码，它锁定和解锁 Application 对象并创建一个新的学生 ID。若 idNum 变量不存在，那么将创建它。若它存在，那么将当前值加 1 并将它赋给 studentID 变量。

```
Application.Lock();
if (!Application.Contents("idNum")) {
        Application.Contents("idNum") = 100;
        var curID = Application.Contents ("idNum");
        Session.Contents("studentID")=curID;
}
else {
        var curID = Application.Contents("idNum");
        ++curID;
        Session.Contents("studentID") = curID;
        Application.Contents("idNum") = curID;
}
Application.Unlock();
```

4．输入下面部分，它打开一个与 WebAdventureASP.dsn 的数据库连接。该代码使用了 ConnectionTimeout（30 秒）并检测是否正确建立连接。

```
var dbConnection = Server.CreateObject("ADODB.Connection");
dbConnection.ConnectionTimeout = 30;
dbConnection.Open("FILEDSN=WebAdventureASP.dsn");
if (dbConnection.State == "adStateClosed")
        Response.Write("The database is not available.");
```

5．添加结束的 ASP 界定符%>。

6．在 Data Disk 的 Tutorial.11 文件夹内将文件存为 GetStudentIDASP.asp。

11.2.3　执行 SQL 命令

在 ADO 中有三种技术被用来执行对数据库的 SQL 查询：Connection 对象的 Execute()方法、Recordset 对象和 Command 对象。本教程讨论了 Connection 对象的 Execute()方法和 Recordset 对象。Command 对象使用更先进的过程，它让能够编译对数据源的查询及使用不同的值集来重复查询。请参阅 Microsoft 的 ASP 文档了解使用 Command 对象的内容。

Execute()方法

Execute()方法向数据库管理系统发送用于处理的 SQL 语句，并且它与 LiveWire 的 execute()方法等价。使用 Execute()方法来更新或修改数据库表。使用 Execute()方法的语法是 database.Execute(SQL statements);。不要使用 Execute()方法提交 ODBC SQL 语句。相反，应以所访问的数据库管理系统的原始 SQL 语言提交 SQL 转移语句。

ADO 的 Execute()方法将查询的结果返回给 Recordset 对象。Execute()方法返回的 Recordset 对象是前向游标并且是只读的。前向游标只能向前移动来遍历结果集中的记录，不能向后或是移动到特定的记录。若需要对 Recordset 对象进行更多的操作，那么需要构造自己的 Recordset 对象（接下来将学习 Recordset 对象）。Execute()方法对快速插入、修改或删除数据库中的行非常有用。例如，下面的代码使用 SQL INSERT 语句将一条新的雇员记录追加到 Employees 数据库表。首先将 SQL 代码赋给 SQLString 变量，然后使用 dbConnection.Execute(SQLString);语句执行它。

```
var dbConnection = Server.CreateObject("ADODB.Connection");
dbConnection.ConnectionTimeout = 30;
dbConnection.Open("FILEDSN=Accounting.dsn");
if (dbConnection.State == "adStateClosed")
        Response.Write("The database is not available.");
var SQLString = "INSERT INTO Employees VALUES('106', 'Mbuti',
'Pierre', '106 Flagg Road', 'Spencer', 'MA', '01562', 'x413')";
dbConnection.Execute(SQLString);
dbConnection.Close();
```

下面的代码演示了 Execute()方法的另一个实例，它从 Employees 表删除一行。

```
var dbConnection = Server.CreateObject("ADODB.Connection");
dbConnection.ConnectionTimeout = 30;
dbConnection.Open("FILEDSN=Accounting.dsn");
if (dbConnection.State == "adStateClosed")
        Response.Write("The database is not available.");
var SQLString = "DELETE FROM Employees
        WHERE Last_Name = 'Miller'";
```

```
dbConnection.Execute(SQLString);
dbConnection.Close();
```

前面代码中的 SQL 字符串使用 WHERE 子句在表中搜索 Last_Name 字段等于“ Miller ”的行。请注意前面的语句将真正地删除表中 Last_Name 字段等于“ Miller ”的所有行。该语句对的意图是安全的，因为知道仅存在一条记录在 Last_Name 字段中包含了“ Miller ”。请确保在执行 DELETE 语句之前，真正理解将要删除哪些记录；确保在使用 DELETE 语句时包括 WHERE 子句，否则将删除指定表中的所有行。

下一步，将向 GetStudentIDASP.asp 文件中添加代码，使用 Execute()方法向数据库中写入记录：

1．返回到文本编辑器或 HTML 编辑器窗口中的 GetStudentIDASP.asp 文件。

2．在结束的 ASP 界定符的前面，添加下面的语句来创建 SQL 字符串和运行 Execute()方法。该语句包含在 else 结构中，当且仅当前面的、用来判断数据库是否连接的 if 语句返回 adStateOpen 值时才执行它。

```
else {
    var SQLString = "INSERT INTO Students VALUES('"
        + curID + "', '"
        + Request.Form("last_name") + "', '"
        + Request.Form("first_name") + "', '"
        + Request.Form("address") + "', '"
        + Request.Form("city") + "', '"
        + Request.Form("state") + "', '"
        + Request.Form("zip") + "', '"
        + Request.Form("email") + "')";
    dbConnection.Execute(SQLString);
```

3．下一步，添加下面的代码，它向客户返回一条包含了最近创建的学生 ID 的响应：

```
Response.Write("<H2>WebAdventure Computer Training
Registration</H2>");
Response.Write("Thanks " + Request.Form("first_name")
    + "! Your new student ID is <B>" + curID + "</B>");
Response.Write(". Click <A HREF='CourseListingASP.asp'>
here</A> to proceed to the course registration page.");
```

4．关闭 dbConnection 和 else 结构。

```
    dbConnection.Close();
}
```

5．保存并关闭 GetStudentIDASP.asp，接着将文件拷贝或上传到 ASP 服务器。

6．在 Web 浏览器内从 ASP 服务器打开 RegistrationASP.html 文件。填充注册信息并单击“ Get Student ID ”按钮。将会收到一条如图 11-19 所示的响应。记下该新的学生 ID，因为在下一个练习中将需要它。

图 11-19　GetStudentIDASP.asp 返回的响应

7．关闭 Web 浏览器窗口。

下一步，创建 RegisterStudentASP.asp 文件：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．输入 ASP 处理命令和起始的 ASP 界定符：

```
<%@ LANGUAGE=JScript %>
<!-- #include file="adojavas.inc" -->
<%
```

3．添加下面的代码打开数据库连接：

```
var dbConnection = Server.CreateObject("ADODB.Connection");
dbConnection.ConnectionTimeout = 30;
dbConnection.Open("FILEDSN=WebAdventureASP.dsn");
if (dbConnection.State == "adStateClosed")
        Response.Write("The database is not available.");
```

4．下一步，添加下面的代码，使用 Execute()方法执行 SQL 语句：

```
else {
        var SQLString = "INSERT INTO Registration VALUES('"
                + Request.Form("id") + "', '"
                + Request.Form("course") + "', '"
                + Request.Form("days") + "', '"
                + Request.Form("time") + "',";
```

```
        dbConnection.Execute(SQLString);
```

5．添加下面的代码，它将一条响应返回给用户。

```
Response.Write("<H2>WebAdventure Computer Training
        Registration</H2>");
Response.Write("You are registered for "
        + Request.Form("course") + " on "
        + Request.Form("days") + ", "
        + Request.Form("time"));
Response.Write(". To register for another course, click
        <A HREF='CourseListingASP.asp'>here</A>. Or click
        <A HREF='ReviewScheduleASP.asp'>here</A> to review
        your current schedule.");
```

6．关闭 dbConnection 和 else 结构，并添加终止的 ASP 命令。

```
        dbConnection.Close();
}
%>
```

7．在 Data Disk 的 Tutorial.11 文件夹内将文件存为 RegisterStudentASP.asp，接着将文件拷贝或上传到 ASP 服务器。在 Web 浏览器内从 ASP 服务器打开 RegisterStudentASP.asp。输入前面创建的学生 ID 并单击“ Class Registration ”按钮打开 CourseListingASP.asp。图 11-20 显示了文档在 Internet Explorer 中的结果。

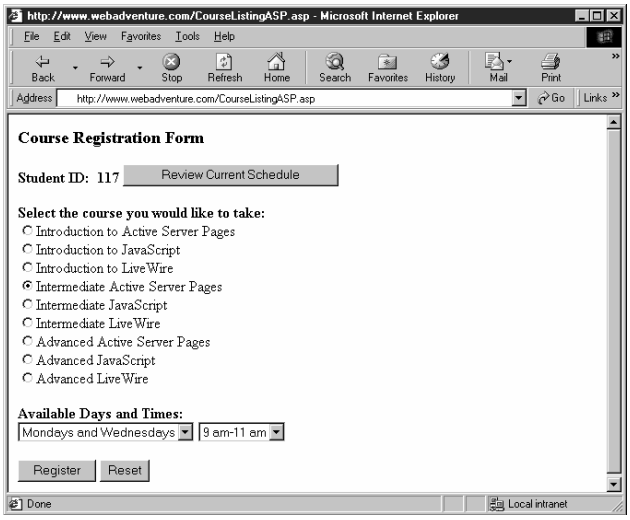


图 11-20　Internet Explorer 中的 CourseListingASP.asp

8．填充课程表单并单击“ Register ”按钮。图 11-21 显示了 RegisterStudentASP.asp 返回的响应的实例。

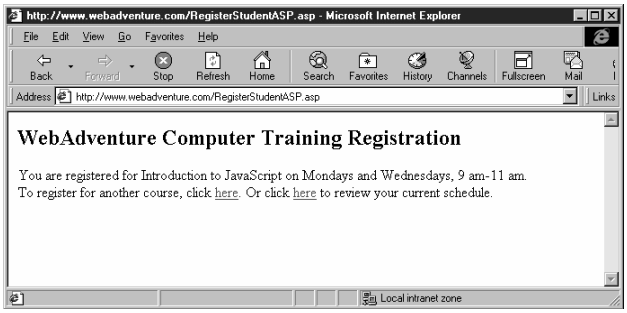


图 11-21　来自 RegisterStudentASP.asp 的响应

9．关闭 Web 浏览器窗口。

Recordset 对象

ADO 包括了 Recordset 对象，它被用来访问、添加、删除和修改数据库记录。可以把 Recordset 对象看作完全与 LiveWire 的 Cursor 对象等价，因为它也返回一个数据库值的虚拟表。与 LiveWire 的 Cursor 对象一样，能够检索值或添加、删除和修改来自 Recordset 对象的虚拟表的记录。任何作出的对 Recordset 对象的虚拟表中记录的修改都将被写入服务器上真正的数据库内。使用与创建 Connection 对象实例的相同方法来创建 Recordset 对象实例：使用 Server 对象的 CreateObject() 方法。然而，请使用 ADODB.Recordset 代替 ADODB.Connection 作为 progID。下面的语句创建一个新的 rsEmployees Recordset 对象：

```
var rsEmployees = Server.CreateObject("ADODB.Recordset");
```

在创建了 Recordset 对象实例之后，必须立即使用 Recordeset 对象的 Open()方法返回一个特定的记录集。使用 Recordset 对象的 Open()方法的语法是 recordset variable.Open();。在使用 Open()方法之前，需要初始化 Recordset 对象的属性来指定连接信息、查询规则和其他选项，Recordset 对象在检索记录集时将使用它们。表 11-9 列出了 Recordset 对象的常用属性。

表 11-9　常用的 Recordset 对象属性

属　性	描　述
ActiveConnection	用来打开 Recordset 对象的数据库连接
BOF	若游标位于文件的开始处，那么它为 true 值；否则为 false
EOF	若游标位于文件的结束处，那么它为 true 值；否则为 false
CursorLocation	它能够被设为三种值中的一种，用来指定在何处管理游标：adUseNone、adUseClient 或 adUseServer。adUseClient 表示客户管理游标；adUseServer 表示服务器管理游标；adUseNone 表示不管理游标

续表

属　性	描　述
CursorType	记录集使用的游标类型：adOpenForwardOnly、adOpenKeyset、adOpenDynamic 或 adOpenStatic
MaxRecords	查询返回的最大记录数
RecordCount	记录集中的记录数
Source	用于检索数据集的 SQL 字符串

下面的实例演示了在使用 Open()方法之前怎样初始化 Recordset 对象的 Source 和 ActiveConnection 属性。ActiveConnection 属性指定了 Recordset 对象用来访问数据库记录的数据库连接。实例首先将 Programmers 数据库作为 dbConnection 打开，使用 dbConnection 对象的 State 属性确定数据库是否可用，接着将 ADODB.Recordset 作为 CreateObject()方法的 progID 打开 rsEmployees 记录。下一步，将一个 SQL 语句赋给 Source 属性，dbConnection 对象被赋给 rsEmployees 对象的 ActiveConnection 属性，接着调用 Open()方法。SQL 语句必须是有效的、用 ODBC SQL 编写的 SELECT 语句。

```
var dbConnection = Server.CreateObject("ADODB.Connection");
dbConnection.ConnectionTimeout = 30;
dbConnection.Open("FILEDSN=Programmers.dsn");
if (dbConnection.State == "adStateClosed")
    Response.Write("The database is not available.");
var rsEmployees = Server.CreateObject("ADODB.Recordset");
rsEmployees.Source = "SELECT * FROM Employees
    ORDER BY Last_Name, First_Name";
rsEmployees.ActiveConnection = dbConnection;
rsEmployees.Open();
additional statements ;
dbConnection.Close();
```

提示：确定在使用 Open()方法之前初始化了 Recordset 对象属性，否则将会收到错误。

除 Open()方法之外，Recordset 对象还包括了其他一些方法和属性。表 11-10 列出了 Recordset 对象一些常用方法。

表 11-10　常用的 Recordset 对象方法

方　法	描　述
AddNew()	创建新记录
CancelUpdate()	取消任何未决的对记录的修改
Close()	关闭 Recordset 对象
Delete()	删除当前记录

续表

方　法	描　述
Move()	移至特定的记录号处
MoveFirst()	移至第一条记录处
MoveLast()	移至最后一条记录处
MoveNext()	移至下一条记录处
MovePrevious()	移至上一条记录处
Open()	为 Recordset 对象打开一个游标
Requery()	通过重新执行查询刷新 Recordset 对象中的数据
Resync()	在不重新执行查询的情况下，刷新 Recordset 对象中的数据
Save()	将 Recordset 对象数据保存在文件中
Seek()	在 Recordset 对象中搜索与特定规则匹配的记录
Supports()	返回 Recordset 对象所支持的功能类型
Update()	保存任何未决的对记录的修改

应该一直使用的 Recordset 对象方法是 Close()方法。在所有 Recordset 对象关闭之前，不能使用 Connection 对象的 Close()方法来关闭数据库连接。下面的代码演示了前面的一个实例，但是使用了 Recordset 对象的 Close()方法。

```
var dbConnection = Server.CreateObject("ADODB.Connection");
dbConnection.ConnectionTimeout = 30;
dbConnection.Open("FILEDSN=Programmers.dsn");
if (dbConnection.State == "adStateClosed")
    Response.Write("The database is not available.");
var rsEmployees = Server.CreateObject("ADODB.Recordset");
rsEmployees.Source = "SELECT * FROM Employees
    ORDER BY Last_Name, First_Name";
rsEmployees.ActiveConnection = dbConnection;
rsEmployees.Open();
additional statements ;
rsEmployees.Close();
dbConnection.Close();
```

提示：请注意在前面的代码中使用了两个 Close()方法：一个用来关闭 Recordset 对象，另一个用来关闭 Connection 对象。

遍历 Recordset 对象：Recordset 对象中的位置称为游标，与使用 LiveWire 的 Cursor 对象时一样。然而，当首次创建一个 Recordset 对象时，游标开始放置在记录集的第一行内，而不是在记录集中第一行前。LiveWire 的 Cursor 对象就是如此。数据库表中的字段名被赋给了 Recordset 对象的 Fields 集合内的变量。例如，若为 Programmers 数据库初始化了一个

Recordset 对象 rsEmployees，那么使用与 rsEmployees.Fields("First_Name")类似的语句来引用 First_Name 字段。为了遍历 Recordset 对象，请使用 MoveNext()方法。下面的代码创建了一个新的 Recordset 对象，接着将游标向前移动了一行。

```
var rsEmployees = Server.CreateObject("ADODB.Recordset");
rsEmployees.Source = "SELECT * FROM Employees
      ORDER BY Last_Name, First_Name";
rsEmployees.ActiveConnection = dbConnection;
rsEmployees.Open();
rsEmployees.MoveNext();
statements ;
rsEmployees.Close();
```

提示：为了简化前面的代码未使用打开数据库连接的语句。

Recordset 对象比 LiveWire 的 Cursor 对象多包含了几个用来浏览记录的方法。除 MoveNext()方法之外，ADO 还包括了其他 4 个浏览方法：Move()、MoveFirst()、MoveLast()和 MovePrevious()。为了使用除 MoveNext()之外的其他任何浏览方法，必须在使用 Recordset 对象的 Open()方法之前设置 CursorType 属性。CursorType 属性指定了结果记录集内所允许的游标类型。能够将 4 种游标类型中的一种赋给 CursorType 属性。表 11-11 列出了 CursorType 属性的有效类型值。

表 11-11 ADO 游标类型

游 标 类 型	描 述
adOpenForwardOnly	前向游标。此游标只让使用 MoveNext()方法在记录集内向前移动
adOpenKeyset	键集游标。允许在记录集内向前和向后移动，但是不允许看到其他用户添加的新记录。能够看到其他用户所做的修改，但是不能访问任何已被其他用户删除的记录
adOpenDynamic	动态游标。允许在记录集内向前和向后移动并且能看到其他用户所做的修改
adOpenStatic	静态游标。允许在记录集内向前和向后移动，但是不允许看到任何其他用户所做的修改

决定使用哪种游标类型取决于需要从应用程序获取的性能。通常，允许能够看到其他用户所做修改的游标将放慢应用程序的速度。最快的游标是 adOpenForwardOnly，因为它仅让在记录集中移动一遍并且不让看到其他用户所做的任何修改。adOpenForwardOnly 是默认的游标类型，若仅需要在记录集内移动一遍，那么它是最佳选择。若想能够在记录集内向前和向后移动，那么需要使用其他游标类型中的一种。下面的代码演示了怎样将游标的类型设为 adOpenStatic，它允许在记录集内向前和向后移动，但是不让看到其他用户所做的任何修改。实例使用 MoveNext()方法移动到记录集中的第二条记录并打印 First_Name 和

Last_Name 的字段值。接着使用 MovePrevious()方法移回第一条记录处并再次打印 First_Name 和 Last_Name 的字段值。

```
var rsEmployees = Server.CreateObject("ADODB.Recordset");
rsEmployees.Source = "SELECT * FROM Employees
      ORDER BY Last_Name, First_Name";
rsEmployees.ActiveConnection = dbConnection;
rsEmployees.CursorType = adOpenStatic;
rsEmployees.Open();
rsEmployees.MoveNext();
Response.Write(rsEmployees("First_Name") + " "
      + rsEmployees("Last_Name"));
rsEmployees.MovePrevious();
Response.Write(rsEmployees("First_Name") + " "
      + rsEmployees("Last_Name"));
rsEmployees.Close();
```

当使用记录集和浏览方法时，不能确定在当前游标位置前或后是否存在另一条记录。另外，也不能确定 SQL 语句是否真正返回一些记录。例如，前面的代码假定在 Employees 表中存在记录，然而，SQL 语句可能根本不返回任何记录。为了确定下一条或上一条记录是否可用，使用 Recordset 对象的 BOF 和 EOF 属性。若游标位于记录集中第一条记录之前，那么 BOF（文件开始）属性返回真；若游标位于记录集中第一条记录处或之后，那么返回假。类似地，若游标位于最后一条记录之后，那么 EOF（文件结束）属性返回真。若游标位于最后一条记录处或之前，那么返回假。下面的代码演示了怎样使用 while 循环在移动游标之前检测 EOF 属性值。只要 EOF 属性不等于 true，那么 while 循环继续执行。

```
var rsEmployees = Server.CreateObject("ADODB.Recordset");
rsEmployees.Source = "SELECT * FROM Employees
      ORDER BY Last_Name, First_Name";
rsEmployees.ActiveConnection = dbConnection;
rsEmployees.Open();
while(rsEmployees.EOF != true) {
      Response.Write(rsEmployees.Fields("First_Name") + " "
            + rsEmployees.Fields("Last_Name") + " lives in "
            + rsEmployees.Fields("City") + ", "
            + rsEmployees.Fields("State") + "<BR>");
      rsEmployees.MoveNext();
}
rsEmployees.Close();
```

下一步将创建 ReviewScheduleASP.asp 文件，它将显示学生已经注册的课程。

创建 ReviewScheduleASP.asp 文件：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．输入 ASP 处理命令和起始的 ASP 界定符。

```
<%@ LANGUAGE=JScript %>
<!-- #include file="adojavas.inc" -->
<%
```

3．添加下面的代码打开数据库连接：

```
var dbConnection = Server.CreateObject("ADODB.Connection");
dbConnection.ConnectionTimeout = 30;
dbConnection.Open("FILEDSN=WebAdventureASP.dsn");
if (dbConnection.State == "adStateClosed")
        Response.Write("The database is not available.");
```

4．添加起始的 else {语句来包括数据库语句。

5．输入 var rsSchedule = Server.CreateObject("ADODB. Recordset");初始化 Recordset 对象。

6．将 SQL 代码赋给 Source 属性和将 dbConnection 赋给 ActiveConnection 属性。接着运行 Open()方法。SQL 将返回的记录限制在那些与 studentID 匹配的记录内。

```
rsSchedule.Source = "SELECT * FROM Registration WHERE
Student_ID = '" + Session.Contents("studentID") + "'";
rsSchedule.ActiveConnection = dbConnection;
rsSchedule.Open();
```

7．添加下面部分，它将响应返回给学生并关闭 rsSchedule 对象。

```
Response.Write("<H2>This is your current schedule</H2>");
while(rsSchedule.EOF != true) {
        Response.Write(rsSchedule.Fields("Course") + ", "
        + rsSchedule.Fields("Days") + ", "
        + rsSchedule.Fields("Time") + "<BR>");
        rsSchedule.MoveNext();
}
rsSchedule.Close();
```

8．关闭 dbConnection 对象和 else 语句，并添加结束的 ASP 界定符。

```
        dbConnection.Close();
```

```
}
%>
```

9．在 Data Disk 的 Tutorial.11 文件夹内将文件存为 ReviewScheduleASP.asp，接着将文件拷贝或上传到 ASP 服务器。在 Web 浏览器内从 ASP 服务器打开 RegistrationASP.html 文件。输入一个已有的学生 ID 并单击“ Class Registration ”按钮。在 CourseListingASP.asp 文档内单击“ Review Current Schedule ”按钮。图 11-22 显示了一个已注册数个班级的学生的输出结果。

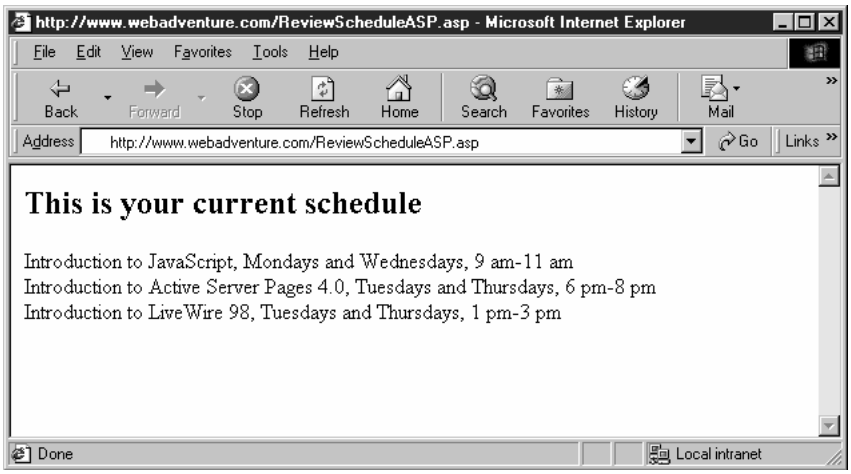


图 11-22　Internet Explorer 中 ReviewScheduleASP.asp

10．关闭 Web 浏览器窗口。

可更新游标：在调用 Open()方法之前使用 Recordset 对象的 LockType 属性来创建一个可更新游标。赋给 LockType 属性的值决定了游标是只读的、悲观的还是乐观的。只读游标禁止用户对记录作出修改。乐观游标仅在执行 Update()方法时才锁定记录。从对一条记录编辑开始，在调用 Update()方法提交此记录之前，悲观游标都禁止其他用户访问此记录。通过将 4 种值中的一种赋给 LockType 属性来确定游标是否是可更新的。表 11-12 列出了 LockType 属性的有效锁定类型。

表 11-12　ADO 锁定类型

所 定 类 型	描　　述
adLockReadOnly	只读
adLockPessimistic	悲观锁定，一条记录接着一条记录
adLockOptimistic	乐观锁定，一条记录接着一条记录
adLockBatchOptimistic	乐观批更新

默认的锁定类型是 adLockReadOnly，它禁止用户对记录做出任何修改。为了对记录做

出修改，必须使用另外三种锁定类型中的一种。若不能判断多个用户同时试图编辑数据库中的同一条记录，请使用乐观锁定。然而，若存在多个用户将同时试图编辑同一条记录的可能性，那么使用悲观锁定更安全，因为它将数据完整性问题减到最小。

可更新游标具有三个特有的 Recordset 对象方法：Update()、Delete()和 AddNew()方法。Update()方法将保存对记录集中当前记录所做的任何修改。在修改了字段值之后执行 Update()方法。例如，假设 Erica Miller 最近刚结婚并将她的姓改为 Lee。下面的代码创建了一个 Recordset 对象，它使用 SELECT 语句和 WHERE 子句来仅检索 Erica Miller 的记录。将 adLockOptimistic 锁定类型赋给 LockType 属性以便记录能够被编辑。在编辑记录之前，代码首先确定 EOF 属性不等于真，以及通过检测 Last_Name 字段值确保记录确实是 Erica Miller 的。在编辑记录之后，将使用 Update()方法更新它。

```
var rsEmployees = Server.CreateObject("ADODB.Recordset");
rsEmployees.Source = "SELECT * FROM Employees
        WHERE Last_Name = 'Miller'";
rsEmployees.ActiveConnection = dbConnection;
rsEmployees.LockType = adLockOptimistic;
rsEmployees.Open();
if (rsEmployees.eof != true) {
        if (rsEmployees.Fields("Last_Name") == "Miller") {
                rsEmployees.Fields("Last_Name") = "Lee";
                rsEmployees.Update();
}
rsEmployees.Close();
```

AddNew()方法与 SQL INSERT 语句类似，因为它将新记录追加到表尾。在执行 AddNew()方法之后，新记录将成为游标的当前位置。接着将新值赋给新记录的字段并执行 Update()方法。下面的代码演示了怎样为 WebAdventure 的新程序员追加新记录：

```
var rsEmployees = Server.CreateObject("ADODB.Recordset");
rsEmployees.Source = "SELECT * FROM Employees";
rsEmployees.ActiveConnection = dbConnection;
rsEmployees.LockType = adLockOptimistic;
rsEmployees.Open();
rsEmployees.AddNew();
rsEmployees.Fields("Employee_ID") = "107";
rsEmployees.Fields("Last_Name") = "Picard";
rsEmployees.Fields("First_Name") = "Lisa";
rsEmployees.Fields("Address") = "1113 Oakham Road";
rsEmployees.Fields("City") = "New Braintree";
rsEmployees.Fields("State") = "MA";
rsEmployees.Fields("Zip") = "01531";
```

```
rsEmployees.Fields("Extension") = "x309";
rsEmployees.Update();
rsEmployees.Close();
```

提示：若在编辑记录时调用 AddNew()方法，那么 ADO 将自动为正在编辑的记录调用 Update()方法，接着创建该新记录。

Delete()方法与 SQL DELETE 语句类似，除了它不是删除所有与 SELECT 语句相匹配的记录，而是仅删除游标所处位置处的记录。例如，若 Scott Morinaga 离开了 WebAdventure，那么可以使用下面的代码来删除他的雇员记录：

```
var rsEmployees = Server.CreateObject("ADODB.Recordset");
rsEmployees.Source = "SELECT * FROM Employees";
rsEmployees.ActiveConnection = dbConnection;
rsEmployees.LockType = adLockOptimistic;
rsEmployees.Open();
while(rsEmployees.Fields("Last_Name")!= "Morinaga") {
        rsEmployees.MoveNext();
}
rsEmployees.Delete();
rsEmployees.Close();
```

11.2.4　ADO 事务处理

在 ASP 中通过 Connection 对象的 BeginTrans()、CommitTrans()和 RollbackTrans()方法来进行事务处理。它们基本上与 LiveWire 的 beginTransaction()、commitTransaction()和 rollbackTransaction()方法等价。使用 BeginTrans()方法开始一个事务，所有跟在 BeginTrans()方法后的数据库语句都是当前事务的一部分。在组成当前事务的数据库语句后，执行 CommitTrans()方法提交对数据库的修改。在 BeginTrans()方法后的任何地方，都能够执行 RollbackTrans()方法来取消组成当前事务的语句所做出的所有修改。例如，若当前事务漏掉了一个字段，如键字段，那么可以调用 RollbackTrans()来取消组成当前事务的所有修改。若在使用 BeginTrans()之前，试图使用 CommitTrans()或 RollbackTrans()，那么将收到来自数据库的一个错误。在响应返回给客户之后，不能再回滚事务。

下面的代码演示在往 Employees 表追加新雇员记录时怎样使用事务处理：

```
var rsEmployees = Server.CreateObject("ADODB.Recordset");
rsEmployees.Source = "SELECT * FROM Employees";
rsEmployees.ActiveConnection = dbConnection;
rsEmployees.LockType = adLockOptimistic;
rsEmployees.Open();
```

```
dbConnection.BeginTrans();
rsEmployees.AddNew();
rsEmployees.Fields("Employee_ID") = "107";
rsEmployees.Fields("Last_Name") = "Picard";
rsEmployees.Fields("First_Name") = "Lisa";
rsEmployees.Fields("Address") = "1113 Oakham Road";
rsEmployees.Fields("City") = "New Braintree";
rsEmployees.Fields("State") = "MA";
rsEmployees.Fields("Zip") = "01531";
rsEmployees.Fields("Extension") = "x309";
rsEmployees.Update();
dbConnection.CommitTrans();
rsEmployees.Close();
```

前面的代码未包括 RollbackTrans()方法。在讨论错误处理时将学习怎样回滚事物。

11.2.5 ADO Error 对象错误处理

所有涉及 ADO 对象的操作都可能产生错误信息。当 ADO 内产生错误信息时，错误对象将被添加到 Errors 集合中。Errors 集合是一列最近 ADO 操作所返回的错误对象。Errors 集合包含了惟一一个属性 Count，它返回在最近 ADO 操作过程中添加到 Errors 集合中错误的个数。若 Count 属性值为 0，那么上一次 ADO 操作成功完成。在返回非 0 值时，能够使用每个 Error 对象的属性来查阅更多的有关错误的内容。Error 对象含有一个包含了错误号的 Number 属性、一个包含了错误描述的 Description 属性、一个标识错误源的 Source 属性以及提供数据源信息的 SQL 和 State 属性。在执行事务出现问题时，能够使用 Count 属性和 Error 对象的属性以及 RollbackTrans()方法来取消任何对数据库的修改。下面的代码演示了一个前面看到过的实例，但是这次使用了 RollbackTrans()方法。代码首先检测 Count 属性，若它等于 0，那么提交事务。若 Count 属性比 0 大，那么回滚事务并执行 while 循环遍历 Errors 集合中的内容，返回每个错误号和描述。

```
var rsEmployees = Server.CreateObject("ADODB.Recordset");
rsEmployees.Source = "SELECT * FROM Employees";
rsEmployees.ActiveConnection = dbConnection;
rsEmployees.LockType = adLockOptimistic;
rsEmployees.Open();
dbConnection.BeginTrans();
rsEmployees.AddNew();
rsEmployees.Fields("Employee_ID") = "107";
rsEmployees.Fields("Last_Name") = "Picard";
rsEmployees.Fields("First_Name") = "Lisa";
```

```
rsEmployees.Fields("Address") = "1113 Oakham Road";
rsEmployees.Fields("City") = "New Braintree";
rsEmployees.Fields("State") = "MA";
rsEmployees.Fields("Zip") = "01531";
rsEmployees.Fields("Extension") = "x309";
rsEmployees.Update();
if (dbConnection.Errors.Count == 0) {
        rsEmployees.CommitTrans();
        Response.Write("The record was added successfully.");
}
else if (dbConnection.Errors.Count > 0) {
        rsEmployees.RollbackTrans();
        Response.Write("The record was not successfully added.
                The database server returned the following error
                number(s) and description(s):<BR>")";
        var curError = 0;
        while(curError < dbConnection.Errors.Count)) {
                Response.Write("Error Number: "
                        + dbErrors.Errors.Error(curError.Number);
                Response.Write("Error Description: "
                        + dbErrors.Errors.Error(curError.Description);
                ++curError;
        }
}
rsEmployees.Close();
```

11.2.6　总结

- ActiveX Data Object（ADO）是一种 Microsoft 数据库连接技术，它让 ASP 和其他 Web 开发工具能够访问与 ODBC 和 OLE DB 兼容的数据库。
- OLE DB 是 Microsoft 提出的为 ODBC 后继的数据源连接标准。
- 在创建需要访问数据库的 ASP 应用程序时，必须将 adojavas.inc 文件拷贝到包含 ASP 文件的目录中并使用#include 命令在程序中插入此文件。
- #include 命令指定了在服务器处理 ASP 文件之前插入到其中的文件。
- ADO 技术基于由对象和集合组成的对象模型来访问和维护数据源。
- ADO Connection 对象用来在 ASP 中访问数据库，在功能上与 LiveWire 的 Database 对象类似。
- 若在过去指定的时间后还未成功连接数据库，Connection 对象的 ConnectionTimeout 属性将放弃客户的数据库连接尝试。
- Execute()方法向数据库管理系统发送用于处理的 SQL 语句。

- ✧ Execute()方法将查询结果返回给 Recordset 对象。
- ✧ ADO 包括了 Recordset 对象，它被用来访问、添加、删除和修改数据库记录。
- ✧ 在创建 Recordset 对象实例之后，必须使用 Recordset 对象的 Open()方法来返回指定的记录集。
- ✧ 在首次创建 Recordset 对象时，游标刚开始放置在记录集的第一行内。
- ✧ CursorType 属性指定了结果记录集内所允许的游标类型。
- ✧ 若游标位于记录集中第一条记录之前，那么 BOF（文件开始）属性返回真。若游标位于记录集中第一条记录处或之后，那么返回假。
- ✧ 若游标位于最后一条记录之后，那么 EOF（文件结束）属性返回真。若游标位于最后一条记录处或之前，那么返回假。
- ✧ 使用 Recordset 对象的 LockType 方法来在 ADO 内创建可更新游标。
- ✧ 默认的锁定类型是 adLockReadOnly，它禁止用户对记录做出任何修改。
- ✧ 可更新游标特有的三个 Recordset 对象方法是：Update()、Delete()和 AddNew()方法。
- ✧ 在 ASP 中通过 Connection 对象的 BeginTrans()、CommitTrans()和 RollbackTrans()方法来进行事务处理。
- ✧ Errors 集合是一列最近 ADO 操作所返回的错误对象。

11.2.7 问题

1．ODBC 和 OLE DB 之间的主要区别是什么？

a．ODBC 仅能在 Windows 平台上使用

b．OLE DB 的功能仅限于关系数据库

c．OLE DB 不仅能访问关系数据库也能访问非关系数据源

d．ODBC 功能仅限于非关系数据库

2．为了让 ASP 访问数据源，哪种 ADO 文件是必需的？

a．adojavas.inc

b．adojavas.dll

c．adoasp.asp

d．adodatasource.exe

3．在 Active Server Pages 内，怎样使用外部文件？

a．利用处理命令包括 SRC 属性

b．将外部文件放置在 ASP 程序目录下

c．使用#include 命令

d．包括 HREF 属性，作为<%…%>ASP 界定符的一部分

4．哪种是在 ASP 中创建数据库对象的正确语法？

a．var dbConnection = Server.CreateObject("ADODB.Connection");

b．var dbConnection = CreateObject("ADODB.Connection");

c．var dbConnection = new ADODB.Connection;

d．var dbConnection = Server ("ADODB.Connection");

5．若在过去指定时间之后未成功连接数据库，那么 Connection 对象的哪种属性将放弃客户的数据库连接尝试？

a．Timeout

b．Abandon

c．Cancel

d．ConnectionTimeout

6．若已成功地连接了数据库，那么 State 属性将返回哪种值？

a．adStateClosed

b．adStateOpen

c．adStateConnected

d．adStateReady

7．哪种 ADO 方法执行 SQL 转移语句？

a．dbExecute()

b．SQL()

c．Execute()

d．execute()

8．传递哪种参数给 CreateObject()方法来初始化 Recordset 对象？

a．ADO.Recordset

b．ADODB.Recordset

c．ADODB.Record

d．ADODB.Cursor

9．哪种方法被用来向前遍历记录集？

a．Move()

b．Next()

c．MoveNext()

d．Forward()

10．下列 CursorType 属性的哪种参数不允许使用 MoveFirst()方法？

a．adOpenForwardOnly

b．adOpenKeyset

c．adOpenDynamic

d．adOpenStatic

11．怎样才能知道已经移至记录集中的最后一条记录？

a．在 MoveNext()属性返回假时

b．在 EOF 属性返回真时

c．在 BOF 属性返回真时
d．在 EOF 和 BOF 都返回真时
12．怎样创建可更新游标？
a．任何时候使用 CreateObject()方法创建的都是可更新游标
b．使用 true 值作为 Open()方法的 updatable 参数
c．将 Recordset 对象的 LockType 属性设置为可更新值
d．不能创建可更新游标
13．下面哪种方法不是可更新游标特有的？
a．Update()
b．AddNew()
c．Delete()
d．MovePrevious()
14．哪种方法取消一个事务？
a．Cancel()
b．Discard()
c．RollbackTrans()
d．RollbackTransaction()
15．Errors 对象的 Count 属性包含何种值时才表示已成功执行了数据库操作？
a．OK
b．SUCCESS
c．1
d．0

11.2.8 练习

1．在本教程注册程序内所创建的的新学生 ID 是根据 Application 对象的 Contents 集合的 idNum 属性生成的。根据 Application 对象的属性生成新学生 ID 并不是创建它的最好方法，因为每次重新启动注册程序时都将重新初始化 idNum 的值。修改 ASP 版的注册程序以便根据数据库中的 Students 表来创建新学生 ID 而不是根据 Application 对象的 Contents 集合。将需要使用 MoveNext()方法遍历 Students 表内的记录以便检索最后一个赋给学生的 ID。对此练习，请使用整个 11.1 节中所使用的 WebAdventureLiveWire DSN。

2．对下面的 4 个练习，请使用能够访问的数据库管理系统（如 Access、Paradox 或 SQL Server）创建将存储数据的数据库文件。请访问 Microsoft 的通用数据访问网址 http://www.Microsoft.com/data 并在 Internet 上搜索与通用数据访问相关的其他信息来源。通用数据访问的对手是谁？业界支持 Microsoft 提出的不断发展的技术程度如何？撰写一篇有关你的发现的论文。

3．创建一个将网址点击次数保存在数据库中的 ASP 程序。

4．创建一个将网址客户簿记录保存在数据库中的 ASP 程序。

5．创建将记录保存到数据库的电话目录应用程序。在数据库中应该包括标准的电话目录字段如姓名、地址、城市、州、邮编和电话号码等。创建一个 HTML 文档作为主“ 目录 ”，其中能够选择和检索记录。请创建一个 HTML 文档，能够使用它往数据库添加记录；以及另外一个 HTML 文档，其中能够编辑记录。请使用任何能够访问的与 ODBC 兼容的数据库（如 Paradox 或 Access）来创建数据库表。

6．用 ASP 为在 11.1 节练习中创建的在线书店重新创建一个购物车应用程序。请在应用程序中为不同类型的书籍使用不同的页面。购物车应列出用户想购买的书籍并提供一种将信息写入数据库和给客户返回响应的“ 校验 ”机制。

第 12 章　使用 Java 小应用程序和嵌入数据

案例

WebAdventure 越来越多的客户希望除了文本和图片外，能够把不同类型的媒体集成到他们的 Web 站点中。某些客户希望使用比 JavaScript 和 HTML 创建的程序功能更强的 Java 小应用程序。其他一些客户希望包含用于娱乐的音频和用于教育和培训的视频。为了满足这些客户的要求，WebAdventure 的经理要求学习如何在 HTML 文档中包含 Java 小应用程序和嵌入数据，以及如何使用 JavaScript 操纵它们。

预览随机数游戏和嵌入数据程序

在本章中，将创建两个项目：一个包含 Java 小应用程序编写的随机数游戏的 HTML 文档和一个包含嵌入数据的 HTML 文档。将学习如何使用 JavaScript 操纵 Java 小应用程序和嵌入数据。

预览随机数游戏程序:

1. 在 Web 浏览器中打开盘上 Tutorial.12 目录中的 Tutorial12_RandomNumberGame.html 文件。程序会提示在 0 到 100 之间猜一个随机选择的数字。试着猜一下这个随机数。如果单击“ Reload ”或者“ Refresh ”按钮，就会产生一个新的随机数。图 12-1 是该文档在 Internet Explorer 中的显示效果。

图 12-1　Internet Explorer 中的 Tutorial12_RandomNumberGame.html

2. 玩过游戏之后，关闭 Web 浏览器然后在文本编辑器或 HTML 编辑器中打开 Tutorial12_RandomNumberGame.html 文件。此文档包含一个用于装载 Java 小应用程序 Tutorial12_RandomNumberGame.class 的<APPLET>标签。脚本部分的 JavaScript 语句使用<APPLET>标签的 NAME 属性调用小应用程序的方法和属性。

3. 在文本编辑器或 HTML 编辑器中打开 Tutorial12_RandomNumberGame.java 文件。这个文件是用于创建 Java 程序的源代码。在本章中，将学习如何创建 Java 程序以及如何使用 JavaScript 操纵它。

4. 关闭文本编辑器或 HTML 编辑器。

接下来，打开包含由 JavaScript 控制的嵌入数据的 HTML 文档。打开这个文档，需要在系统中安装 RealPlayer。RealPlayer 是一个免费程序，它可以播放多媒体文件，比如视频和音频文件。该程序作为一个独立程序安装在计算机中。它还同时安装在 Web 页面中直接播放多媒体文件的插件和 ActiveX 控件。运行本章的练习需要安装 RealPlayer 7。可以从网址 http://www.real.com/player/index.html 下载 RealPlayer G2。

提示：ActiveX 控件是一个特殊的对象，可以把它称为“迷你应用程序”。可以在 Web 页面或者其他类型的程序中使用 ActiveX 控件。在本章的第 2 节将学习有关 ActiveX 技术的知识。

预览包含由 JavaScript 控制的嵌入数据的 HTML 文档：

1．如果使用的是 Navigator，在浏览器中打开盘上 Tutorial.12 目录下的文档 Tutorial12_RealPlayerPlugin.html。如果使用的是 Internet Explorer，则打开文档 Tutorial12_RealPlayerActiveX.html。这两个程序都会显示一段视频剪辑，可以使用执行 JavaScript 代码的表单按钮来启动、暂停和运行它。试着播放、暂停和停止播放这个视频剪辑。图 12-2 显示了 RealPlayerPlugin.html 在 Navigator 中如何显示。

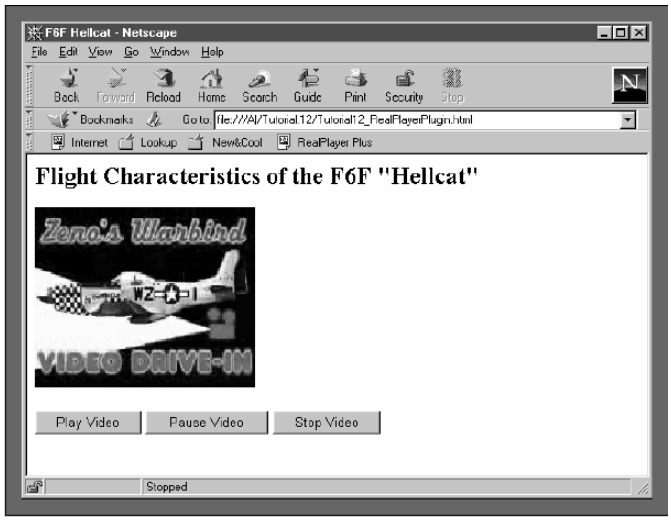


图 12-2　Navigator 中的 Tutorial12_RealPlayerPlugin.html

提示：RealPlayer 播放的视频剪辑是几秒钟的第二次世界大战的 F6F“悍妇”战斗机的教育视频。这段视频剪辑可以从 Zeno 的视频电影网站 Warbird 上获得，该网站上有许多老式军用战斗机的 RealPlayer 视频。如果想看其他的 F6F 视频，或者其他老式军用战斗机的视频，可以通过网址 http://www.zenoswarbirdvideos.com/访问 Zero 的视频电影网站 Warbird。这个网站对于展示如何在 HTML 文档中集成多媒体是一个非常优秀的范例。

2. 看完这个视频之后，关闭 Web 浏览器并在文本编辑器或 HTML 编辑器中打开 Tutorial12_ RealPlayerPlugin.html 和 Tutorial12_RealPlayerActiveX.html 文件。浏览一下 Tutorial12_ RealPlayerPlugin.html 和 Tutorial12_RealPlayerActiveX.html 文件中的标签，会看到它们使用不同的标签来显示嵌入的 RealPlayer 视频。Navigator 中插件文件 Tutorial12_ RealPlayerPlugin.html 使用的是<EMBED>标签。Internet Explorer 中使用的 ActiveX 文件 Tutorial12_RealPlayerActiveX.html 使用<OBJECT>和<PARAM>标签。将在第 2 节学习有关插件和 ActiveX 控件的知识。

12.1 Java 教 程

本节目标

在本节将学习：

- ✧ 关于小应用程序和嵌入数据
- ✧ 关于 Java 类和方法
- ✧ 如何编译 Java 程序
- ✧ 如何创建小应用程序
- ✧ 关于 Java 变量和数据类型
- ✧ 如何在 HTML 文档中添加小应用程序
- ✧ 如何使用 JavaScript 控制小应用程序

12.1.1 小应用程序和嵌入数据

Web 页面不仅由 HTML 标签、图片和 JavaScript 代码组成，还可以包括小应用程序和视频或音频文件这样的嵌入数据。小应用程序是在 Web 页面中运行的 Java 程序。嵌入数据指的是存储在另外一类应用程序中的应用程序数据。Navigator 使用称为插件的特殊程序扩展来显示嵌入数据，Internet Explorer 使用 ActiveX 控件显示嵌入信息。为了在 JavaScript、Java 和插件彼此之间进行通信，Navigator 使用了一项称为 LiveConnect 的技术。Internet Explorer 使用 ActiveX 脚本在 JavaScript 与 Java 和 ActiveX 控件进行交互。Web 页面不同元素之间的这种通信使程序员对页面功能和与用户的交互具有了超凡的控制能力。例如，如

果在 HTML 文档中嵌入了一个计算抵押付款的 Java 小应用程序，就可以使用表单域来为小应用程序的属性赋值以及执行小应用程序的方法。类似地，可以在 HTML 文档中包含允许用户启动和停止插件或 ActiveX 控件运行的控制。控制 Java 程序和 ActiveX 组件的语句可以是（而且通常都是）小应用程序、插件或 ActiveX 控件的一个部分。

本节阐述了 Java 程序设计以及如何使用 JavaScript 操纵小应用程序的基本概念。尽管 Navigator 和 Internet Explorer 使用不同的技术来使用 JavaScript 操纵小应用程序（Navigator 使用 LiveConnect，Internet Explorer 使用 ActiveX），本节中讲解的技术可以在两种浏览器中互换使用。在第 2 节，将学习如何在 Java 中控制 JavaScript 以及如何使用 Navigator 插件和 Internet Explorer 中的 ActiveX 脚本。

12.1.2　Java 介绍

JavaScript 和 Java 是两种截然不同的语言。Java 是 Sun 微系统公司开发的一种面向对象的编译型编程语言。Java 比 JavaScript 要难掌握的多。Java 程序可以作为一个独立的应用程序单独运行，也可以作为一个小应用程序在 Web 页面中运行。相反地，JavaScript 程序仅在 Web 页面中运行，并且只用来控制 Web 页面和 Web 浏览器。Java 可以用来创建 JavaScript 所能创建的所有程序，甚至可以创建比 JavaScript 更多的程序，因为它包含了更多的编程特点、方法和属性。用 Java 语言，可以创建图形和新类型的控制，执行一些网络功能，与开发完善的用户交互界面。还可以利用一种称为多线程的技术创建比 JavaScipt 功能更强大的动画。

既然 Java 是如此的强大，那么为什么还要使用 JavaScript 呢？JavaScript 可以在 HTML 页面的范围之内工作。虽然这个特性限制了 JavaScript 无法创建独立于 Web 页面的应用，但是它保证了 JavaScript 可以和 HTML 文档的各种元素紧密地结合。例如，就无法将 Java 代码直接插入到 HTML 文档中。只能将 Java 的小应用程序以一个有边框的封装对象插入到 HTML 文档中。封装表示所有的代码和数据都包含在对象自身之中。边框是 Web 页面中的一块矩形区域，一个小应用程序可以在该区域中运行。如果需要和封装对象交互，必须运用 LiveConnect 来存取对象的方法和属性。LiveConnect 为 Navigator 提供了存取一个小应用程序的方法和属性的能力，同时为小应用程序提供了操作页面中的 JavaScript 的能力。

提示：与 Java 编程有关的网站是如此的繁多，以至于没有可能将它们一一列出。可以找到提供专用或免费的小应用程序的各种站点，以及介绍 Java 编程和共享 Java 程序和技术的站点。如果需要搜寻这样一个站点，请在 Yahoo!或其他 Internet 搜索引擎中输入 Java 或 applets。其中一个较受欢迎的 Java 编程站点是 Gamelan，该站点自封为“ Java 的官方目录 ”，因为该站点是 Sun 微系统公司支持的。该站点的地址是 http://www.gamelan.com。

提示：最初 JavaScript 称为 LiveScript。在 Navigator 2.0 发布时，为了利用逐渐广泛应用的 Java 语言的影响，改称为 JavaScript。

既然可以用 Java 实现所有的功能，为什么还要用 JavaScript 来控制 Java 小应用程序呢？

因为必须让一个 Web 页面中的所有组成部分（JavaScript 代码、HTML、小应用程序以及插件）紧密结合，像一个集成的程序一样一起工作。理解“ 集成的程序 ”的含义，有助于更好地了解面向对象编程。

面向对象编程 Object-oriented Programming（OOP）指创建可重新利用的软件对象可以容易地在另一个程序中使用。对象指可以当作一个单独的个体或组件的程序代码和数据。对象的范围从简单的控件（例如一个按钮）到整个的程序，例如一个数据库应用。OOP 可以让程序员重新利用自己以前编写的或其他程序员编写的程序对象。在 Java 编程中，一个小应用程序代表一个对象。

小应用程序作为一个可编程的对象，必须能够在其他的编程语言中控制它们。在这种情况下，JavaScript 和 HTML 文档代表了主程序，其中将插入一个小应用程序对象。在某些情况下，小应用程序可能是一个自包含的应用，例如抵押计算应用程序。而在另一些情况下，可能只想用小应用程序增强 JavaScript 程序的功能。例如，可能创建了一个包含收集经济信息的表单的 Web 页面，该表单将被传送到一个 CGI 脚本。在表单被提交之前，可以用一个 Java 小应用程序对表单数据进行高级的计算，可以使用在 JavaScript 中没有的一些方法。为了这个原因，需要可以调用小应用程序的方法，并给其传送必须的表单数据。

Java 有别于 JavaScript 的另一个方面是它体系上的中立性，或称可移植性，意味着它可以在任何平台上运行。平台包括一个操作系统和其硬件类型。例如，在 PC 上运行的 MS-DOS 和 Windows 95/98/NT、在 Macintosh 机器上运行的 Mac OS/8 和在 SPARC 上运行的 Solaris 就是不同的平台。在 Java 和 C 开发出来之前，程序只能为一种平台来编写。要在另一种平台上运行同样的程序，必须重新设计和编写程序。Java 消除了为不同的平台重新设计程序的必要。Java 是通过一个称为 Java 虚拟机的特殊语言解释器来实现该功能的。Java 虚拟机（VM）是 Java 编程语言的解释器。Java 语言为其支持的每个平台提供了不同的解释器。要执行一个 Java 程序，用户只需一个他们所用平台的 Java 虚拟机的拷贝。当一个 Java 程序作为一个小应用程序在 Web 页面中运行时，Java 虚拟机包含在 Web 浏览器中，这就消除了随着程序发布 Java 虚拟机拷贝的必要。

提示：可以将 Java 虚拟机想象成一个通用的翻译器。如果 Java 虚拟机解释的是语言，而不是 Java 代码，则它完成的工作是，将文档从一种语言翻译到另一种语言。可以用的语言给世界上的任何人写一封信。由于有 Java 虚拟机进行翻译，收到信的人就可以用他们当地的语言来阅读了。

虽然 JavaScript 和 Java 不是相同的语言，这并不意味着它们并没有相似之处。两者都共享许多相同的语法和语言结构，因为它们都是从 C 语言演变而来。在开始用 Java 工作时，会意识到，许多 Java 的语法和结构在本书的前面章节中已经学习过。需要注意的是，本节只是讲解 Java 的基础知识，以便理解 Java 和 JavaScript 是如何一起工作的 。

提示：如果喜欢学习更多的 Java 编程的知识，可以参考 Joyce Farrell 编写的 Java 编程，该书由 Course Technology 出版。

在尝试该节的练习时，需要拷贝一份 Sun 微系统公司的 Java 开发工具。Java 开发工具

Java Development Kit（或称 JDK），是最初创建 Java 程序的开发环境，并且可以在几种平台上使用，例如，Windows NT/95/98、Solaris、Macintosh。由于 Java 的体系上的中立性，在任何一个平台上的 JDK 中创建的程序可以在其他任何平台上运行。本章中的例子是用 Java 2 创建的，可以在 http://sun.java.com 中下载 JDK 的最新免费版本。下载后，请按照屏幕上的指导将 JDK 安装到硬盘中。

12.1.3　类

在面向对象的编程中，类是方法、属性和数据的集合。所有的 Java 程序都以类开始生存周期。与在一个文档中创建函数不同（正如在 JavaScript 中实现的），在 Java 中，在类中创建方法。方法是包含了一个类的语句和过程的结构，类似于一个函数。通过添加方法、变量，以及将其他类添加到一个新的 Java 类中，来创建一个 Java 程序。

图 12-3 显示了一个名为 HelloWorld 的 Java 类的开始。在 HelloWorld 类的第一行包括了关键字 public 和 class。Java 关键字 class 定义 HelloWorld 为一个类，关键字 public 是一个类的修饰语。一个类修饰语决定了允许使用一个类时的存取类型。其他的类修饰语还包括 abstract 和 final。在本章中，只使用 public 类修饰语。注意，在类的第一行代码后跟着一对表示类的开始和结束的大括号。

```
public class HelloWorld{
}
```

图 12-3　类的结构

Java 文件以一个文本文件创建，文件后缀为.java。当保存一个 Java 文件时，文件的名称必须和类的名称匹配。例如，在图 12-3 中的类 HelloWorld 对应的文件名称必须是 HelloWorld.java。和 JavaScript 一样，Java 是对大小写敏感的，所以 HelloWorld 类和 HelloWorld.java 文件的名称必须匹配。

创建 Java 程序的方法包括在类包中。可以将 Java 类和 JavaScript 对象对比一下，例如窗口对象和文档对象。包或者类库，是一个相关类的集合。标准的 Java 类包包括 java.lang、java.awt、java.applet，以及其他的一些类库。一些诸如 java.lang 的包，包含了许多基本的 Java 方法类型，这些包是自动地引入每个 Java 程序的。其他的包和类必须专门引入需要使用它们的类中。可以在一行语句中引用整个类名，或者在程序中引入包含了类的包和类库。通过在程序的第一行，添加一条后跟类和包名称的 import 语句，可以将需要使用的包或类库引入程序。引入语句的语法是：import package or class;。例如，java.awt 包包含了许多有用的图形用户交互组件，诸如按钮、复选按钮和菜单。在 java.awt 中包括一个标签类，该类用来创建文本标签。如果要在程序中使用标签类，需要在程序的第一行中包括语句 import java.awt.Label;。还可以在语句中使用星号，作为通用符引入一个整个的包。如果要将整个

的 java.awt 包引入一个程序，可以用语句 import java.awt.*;。注意，所有的 Java 语句必须以分号结束。图 12-4 显示了一个 HelloWorld 类的改进版本，其中引入了 java.awt 包。

```
import java.awt.*;
public class HelloWorld{
}
```

图 12-4　引入了 java.awt 包的 HelloWorld 类

提示：可以在 JDK 文档中找到包和类库的完整信息。

12.1.4　方法

在类定义的括号中，放置 Java 程序的方法。在一个类中创建的方法依赖于是创建一个独立的 Java 程序还是一个小应用程序。如果是一个独立的 Java 程序，在类的定义中必须包括一个 main()方法。该 main()方法在独立的 Java 程序运行一开始自动地调用。可以在 main()方法中包含程序中所有的代码，或者在 main()方法中调用自己的用户方法。main()方法的开头必须如下所示：public static void main(String args[])。关键字 public 表示该方法是全局性的，可以在该类之外加以应用。注意，在本章中，只使用 public 方法。关键字 static 表示该程序只在内存中保留一份拷贝，不管该类运行了多少实例。关键字 void 表示该方法没有返回值。和 JavaScript 不同，Java 方法必须返回一个值，或者用 void 标识该方法没有返回值。在方法开头的(String args[])部分用来接收传入该方法的字符串参数。图 12-5 显示了一个有 main()方法的 HelloWorld 类的例子。

```
public class HelloWorld{
    public static void main (String args []) {
        //statements go here
    }
}
```

图 12-5　包含了 main()方法的 HelloWorld 类

提示：注意，图 12-5 使用了和 JavaScript 同样的注释符号//。还可以用/* statements */对语句进行块注释。

下面，将创建一个简单的 Java 程序，该程序往屏幕上打印若干行文本。将使用到 System.out 类，该类作为 java.lang 包的一部分引入 Java 程序。System.out 类包含了两个方法：print()和 println()方法。这两个方法用来在命令行环境下（例如 MS-DOS），将成行的

文本打印到屏幕上。这两个方法和 JavaScript 中的 write()和 writeln()方法相当。与 write()和 writeln()方法一样，print()和 println()方法的惟一不同点在于 println()方法在文本行后面添加一个回车符。这两个函数都接受一个文本字符串或者变量作为参数，对应的语法是 System.out.print(string or variable)；。举例来说，要打印 Hello World，对应的语法应如下所示：System.out.print("Hello World.");。

提示：与 JavaScript 的 writeln()方法不同，println()方法并不需要任何容器元素，例如一个<PRE>…</PRE>标签对。

创建一个简单的使用 print()和 println()方法打印文本的 Java 程序：

1．打开文本编辑器，并创建一个新的文档。

2．输入下面的类的开始代码和开始的括号。

```
public class FirstJavaProgram {
```

3．输入 main()方法的开始代码。

```
public static void main (String args []){
```

4．然后输入下面的两行打印文本的代码。

```
System.out.println("This is the first line in my Java
file.");
System.out.print("This is the second line in my Java
file.");
```

5．添加两个结束括号，一个结束 main()方法，另一个结束类的定义。

6．保存文件为 FirstJavaProgram.java，保存于 Tutorial.12 文件夹中。在运行程序前必须编译程序。

12.1.5　编译 Java 程序

创建了一个 Java 程序后，.java 文件应该用 JDK 中的 javac 程序进行编译。该 javac 程序，就是 Java 的编译器，是一个将.java 文件转换成扩展名为.class 的命令行编译程序。当运行 Java 编译器时，必须输入 Java 程序，并且要按照正确的大小写，后面跟着后缀.java。举例来说，要编译 HelloWorld.java 文件，必须输入 javac HelloWorld.java。一旦一个 Java 程序编译成.class 文件，就意味着它转换成了字节码。字节码是经过编译的 Java 程序，由 Java 虚拟机来执行。当分发一个 Java 程序时，只需分发.class 文件，除非想让用户看到源代码，如果是这样的话，就必须包含.java 文件。在 JDK\bin 目录下的 Java 程序就是适用于

平台的执行编译功能的 Java 虚拟机。当运行 Java 编译器时，JDK\bin 必须保证在 PATH 语句中，或者保证输入了完整的 bin 目录的路径。例如，如果在目录 c:\jdk1.2.2 下安装了 JDK，那么必须包括 c:\jdk1.2.2\bin 目录在 PATH 语句中。如果没有添加 bin 目录就运行编译程序，必须在命令行中输入完整的路径，例如 c:\jdk1.2.2\bin\javac HelloWorld.java。

下面编译并运行 FirstJavaProgram.java 文件：

1．进入系统的命令行状态。命令行的操作可能由于系统的不同有差异。在 Windows 95/98/NT 下，在开始菜单下选择运行，并输入命令。

2．改变路径到数据盘上的 Tutorial.12 文件夹。

3. 如果 JDK 的 bin 目录已经是 PATH 语句的一部分，那么就输入 javac FirstJavaProgram.java 然后按下回车键编译程序。如果没有在 PATH 语句中，那么就必须输入 Javac 程序的完整路径。一定注意文件的大小写正确，并要带着.java 后缀。如果文件正确地编译了，则会有一个新的命令行提示符显示在屏幕上。可以用 dir 命令列一下 Tutorial.12 文件夹中的文件。会看见几个文件，包括新编译完的 FirstJavaProgram.class 文件。

帮助：如果在编译时收到一个错误的消息，检查在 java 源代码中是否发生了大小写错误的情况。检查是否每个语句后面都包括一个分号。并且要保证在代码中的类名称和.java 文件的名称匹配。改正完错误后，确定重新保存文件并重新编译。

4．要运行程序，输入 java FirstJavaProgram。不要包括后缀.class。屏幕将会如图 12-6 所示。

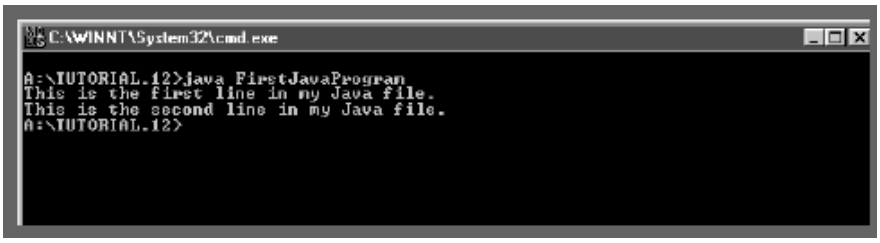


图 12-6　FirstJavaProgram 的输出

5．关闭命令行窗口。

帮助：在 Windows 系统中，可以通过输入 exit 并按下回车键来关闭命令行窗口。

12.1.6　创建一个小应用程序

小应用程序和独立的 JavaScript 应用不同，小应用程序不使用 main()方法。作为替代，小应用程序使用四个方法：init()、start()、stop()和 destroy()。这四个方法在一个小应用程序执行的不同时间点自动地运行。在一个小应用程序第一次打开时，执行 init()方法。在程序从非活动状态转到激活状态时，执行 start()方法。当一个小应用程序停止执行时，执行 stop()

方法。当包含小应用程序的浏览器被关闭时，执行 destroy()方法。会为每个方法编写自己的代码来处理小应用程序的执行。例如，如果小应用程序是一个动画序列，可以用 start()方法在浏览器窗口被激活时重新启动动画。在一个 Java 应用程序中，这些方法不是必须使用的。相反地，可以创建自己的方法，并用 Java 或 JavaScript 控件来执行它们。

要创建一个小应用程序，必须往一个类中引入 java.applet 包。许多小应用程序还引入 java.awt 包，因为该包中包括了许多用来创建小应用程序的图形用户交互组件。还要将 extends Applet 添加在类名称的右边。用 HelloWorld 类作为例子，类的开头语法应该为 public class HelloWorld extends Applet。关键字 extends 允许在一个类的方法、属性的基础上创建一个新类。用来控制一个小应用程序执行的 init()、start()、stop()、destroy()方法，是继承自 java.applet 包中的 Applet 类。

另一个对每个小应用程序都可用的方法是 paint()方法。paint()方法在浏览器的窗口中显示小应用程序的可见组件，并且在窗口最小化、最大化和调整大小时自动的运行。还可以调用自己的 paint()方法。但是，除了直接调用 paint()方法，还可以通过 repaint()方法来调用。paint()方法的开头代码为 public void paint(Graphics g)。括号中的参数创建了一个名为 g 的图形类。图形类用来往小应用程序的边框中插入可视的元素。在本章中常用的一个图形类的方法是 drawString()，它往一个小应用程序的边框中添加字符串文本。drawString()方法的语法是 g.drawString(text,x-coordinate,y-coordinate);。参数中的 x 和 y，表示从小应用程序的边框的左上角开始的相对坐标，文本将被放置在 x 和 y 表示的坐标处。

提示：可以给一个图形对象命名任何名字，但是通常用 g。

图 12-7 显示了一个有五个方法的小应用程序类的基本结构。除了 start()方法，所有的方法都是空的，start()方法调用了 repaint()方法和 paint()方法，以便往小应用程序中插入文本 HelloWorld。

下面开始创建随机数的小应用程序：

1．打开文本编辑器，并创建一个新的文档。

2．输入下面的引入语句，对这个小应用程序来说，这些语句是必须的。

```
import java.applet.*;
import java.awt.*;
```

3. 下面输入类的开头代码。

```
public class RandomNumber extendsApplet{
```

4. 输入下面的代码来创建一个 start()方法，该方法调用 repaint()方法。

```
public void start(){
repaint();
```

}

```
import java.applet.*;
import java.awt.*;
public class HelloWorld extends Applet{
    public void init(){
        //statements go here
    }
    public void start(){
        repaint();
    }
    public void stop(){
        //statements go here
    }
    public void destroy(){
        //statements go here
    }
    public void paint(Graphics g){
        g.drawString("HelloWorld",30,30);
    }
}
```

图 12-7　小应用程序的类结构

5．输入下面的 paint()方法，该方法用 drawString()方法往小应用程序中添加一个名为 displayText 的变量。该变量给用户显示不同的消息，消息依赖于用户的回答和单击的按钮。

```
public void paint(Graphics g) {
    g.drawString(displayText, 30, 30);
}
```

6．添加类的结束括号（}）。

7．保存文件于 Tutorial.12 文件夹中，文件名称为 RandomNumber.java。

12.1.7　Java 变量和数据类型

就像在第 3 章中学习到的，Java 与许多编程语言一样，需要说明一个变量的数据类型。像这种需要说明变量的数据类型的编程语言称为强类型要求的编程语言。而不需要说明变量类型的语言通常称为弱类型要求的编程语言。JavaScript 属于后者，而 Java 属于前者。在 JavaScript 中不允许说明变量的数据类型。由 JavaScript 的解释器自动地决定变量是哪种数据类型。相反地，在 Java 中要说明一个变量，必须指定它的数据类型。Java 支持的基本的数据

类型列在表 12-1 中。需要注意的是，Java 中的数据类型和 JavaScript 中的数据类型并不是精确地对应的。例如，在 Java 中包含了六种数字数据类型，而在 JavaScript 中只有两种。在本章的后面，将阐述在需要的时候，如何使用 Java 和 JavaScript 处理对方的数据类型。

表 12-1　Java 基本数据类型

数 据 类 型	描　　述	实　　例
boolean	逻辑值真或假	true 或 false
byte	8 位整数	值在-128 和 127 之间
char	括在单引号内的任何字符或一个数字 Unicode 字符	'A'、'B'、'C'等。字符 A、B、C 用 Unicode 表示分别是 65、66 和 67
double	64 位浮点数	值在 5e-324 和 1.7976931348623157e+308 之间
float	32 位浮点数	值在 1.4e-45f 和 3.4028235e+38f 之间
int	32 位整数	值在 -2147483648 和 2147483647 之间
long	64 位整数	值在-9223372036854775808 和 9223372036854775807 之间
short	16 位整数	值在-32768 和 32767 之间

与 JavaScript 一样，可以在声明 Java 变量时给它赋值，也可以在后面代码中给它赋值。在变量名的前面包括数据类型名来指明数据类型。可以将数据类型看作代替用来声明 JavaScript 变量的 var 关键字。下面的代码演示了怎样在 Java 中声明数据类型的变量：

```
int integerVariable = 100;
boolean trueOrFalse = true;
char unicodeVariable = 'A';
```

可能已经注意到了没有 string 数据类型。在 Java 中，文本字符串赋给 String 类实例化对象，而不是像在 JavaScript 那样被赋给 string 数据类型。Java 不仅需要声明数据类型，而且还需要声明对象。创建新的 String 对象的语法是 String objectName = new String("text");。在语句的左侧，String 是变量的类名。在语句的右侧，new 关键字使用 String 类的 String() 方法来创建新 String 对象。String 方法传递一个将存储在对象中的文本参数。下面的代码演示了怎样将 Hello World!赋给 helloString String 对象。

```
String helloString = new String("Hello World!");
```

提示：还可以使用简单的形式：String objectName = "text";来创建 String 对象。

可以在类级别（方法外）或方法内声明变量。在方法内声明的对象称为局部变量，与在 JavaScript 函数内声明的局部变量一样。在类级别声明的变量称为实例变量，它与在<SCRIPT>…</SCRIPT>标签对内、函数外声明的全局 JavaScript 变量类似。在 Java 内使用

实例变量术语来引用对象的属性。为了在类外使用实例变量（依据 JavaScript 而言），它必须具有 public 访问修饰符。图 12-8 演示了一个在类外可使用的 String 变量，因为它是在类级别声明的实例变量并且具有 public 访问修饰符。

```
import java.awt.*;
import java.applet.*;
public class HelloWorld extends Applet{
        public String helloString = new String("Hello World");
        public void paint(Graphics g) {
                g.drawString(helloString, 30, 30);
        }
}
```

图 12-8　具有 public 实例变量的 HelloWord 类

下一步，将为 Random Number 程序添加三个实例变量：randomNumber、displayText 和 guessCount。randomNumber 变量是一个整数，用来存储随机产生数。displayText 变量是 String 对象，用来为用户显示不同的消息，它取决于用户的响应和他们单击的按钮。guessCount 变量是一个整数，用来存储用户猜测随机数的次数。惟一需要作为 public 的变量就是 randomNumber 变量，因为以后在 JavaScript 中将它作为 Random Number 应用程序对象的属性来访问。

为 Random Number 程序添加三个变量来存储随机数、所显示的消息和猜测次数的变量：

1．返回到文本编辑器内的 RandomNumber.java 文件。

2．在 start()方法的首部添加下面的变量：

```
public int randomNumber;
String displayText = new String();
int guessCount = 0;
```

3．保存 RandomNumber.java 文件。

对 JavaScript 程序员而言，Java 编程最令人困惑的一个方面是：在 Java 中变量的数据类型在程序执行过程中不能改变。若试图为变量赋给不同的数据类型，那么将会产生错误。若需要将变量的内容作为不同数据类型使用，那么必须将变量强制转换为新的数据类型。强制转换将含在变量内的值拷贝到不同数据类型的变量中。在所需转换的变量的前面添加用括号括起来的目标数据类名字和将值赋给目标类型的变量来转换变量。强制转换变量的语法是 variable = (new_type) old_variable;。下面的代码将整型变量 intNumber 转换为浮点变量 floatNumber：

```
int intNumber = 100;
float floatNumber;
floatNumber = (float) intNumber;
```

下一步，为 Random Number 程序添加创建随机数的代码，并将它赋给 randomNumber 实例变量。为了创建随机数，将使用 Math 类的 random()方法。Java 的 Math 类包含了各种数学属性和执行算术运算的方法。例如，Math 类的 abs()方法返回一个数的绝对值，PI 属性返回 pi（π），它表示圆周长与其半径的比率。Math 类的 random()方法返回一个介于 0.0 和 1.0 之间的双精度数。使用 variable = Math.random();来生成随机数并将它赋给变量。为了将随机数转换为 0 和 100 之间的值，将随机数乘以 100，接着使用 Math 类的 round()方法将其四舍五入为一个整数。round()方法返回一个长整型数。使用 round()方法的语法是 variable = Math.round(number);。因为 randomNumber 实例变量是一个 int 数据类型，需要转换 round()方法所返回的长整数。

提示：JavaScript 含有一个内部的 Math 对象，它包括了在 Java Math 类中发现的许多方法和属性。请参阅附录中的 JavaScript Math 对象属性和方法列表。

为 Random Number 程序添加创建随机数的代码：

1．返回到文本编辑器内的 RandomNumber.java 文件。

2．在 start()方法内的 repaint()方法前面添加下面生成随机数的语句。第一条语句将 random()方法的结果乘以 100 并将它赋给一个双精度变量 doubleNumber。第二条语句使用 round()方法将 doubleNumber 四舍五入为整数，接着将新值赋给长整型变量 longNumber。第三条语句将 longNumber 变量转换为整型并将它赋给 randomNumber 实例变量。

```
double doubleNumber = Math.random() * 100;
long longNumber = Math.round(doubleNumber);
randomNumber = (int) longNumber;
```

3．在将 longNumber 转换为整数和将它赋给 randomNumber 变量的语句后，添加 displayText = "The random number is " + randomNumber +".";。paint()方法在首次载入应用程序时显示 displayText 变量。

4．保存 RandomNumber.java 文件。下一步，切换到命令提示符下，在 Data 磁盘上的 Tutorial.12 文件夹内使用 javac 程序编译 RandomNumber.java。若看到任何编译错误，请确定和更正错误，接着保存此程序并重新编译它。下一步，将学习怎样在 HTML 文件中使用小应用程序。

帮助：若对编译程序有疑难问题，请记住 Java 是区分大小写的并且所有的语句必须以分号结束。还请确认在类首部语句 public class RandomNumber extends Applet {内的 RandomNumber 类名与赋给 RandomNumber.java 文件名的大小写相同。

12.1.8　为 HTML 文档添加 Applet

在使用 Java 编译器编译小应用程序之后，请使用<APPLET>…</APPLET>标签对将它

添加到 Web 页内。<APPLET>标签指示将小应用程序包括在 HTML 文档内。可以使用 4 个属性来创建<APPLET>标签：CODE、WIDTH、HEIGHT 和 NAME。CODE 属性指定了小应用程序类文件。WIDTH 和 HEIGHT 属性决定了包围小应用程序框的尺寸。NAME 属性用来在 JavaScript 内管理小应用程序。图 12-9 包含了一个简单的 HTML 文档，它载入图 12-8 内的 HelloWorld 小应用程序。图 12-10 显示了 HTML 文档和在 Navigator 内小应用程序的样式。含有文本 Hello World!的灰色区域表示小应用程序的围框。

```
<HTML>
<HEAD>
<TITLE>Hello World Applet</TITLE>
</HEAD>
<BODY>
<APPLET CODE="HelloWorld.class" NAME="helloApplet"
WIDTH=100 HEIGHT=50>
</APPLET>
</BODY>
</HTML>
```

图 12-9　含有 HelloWorld 小应用程序的 HTML 文档

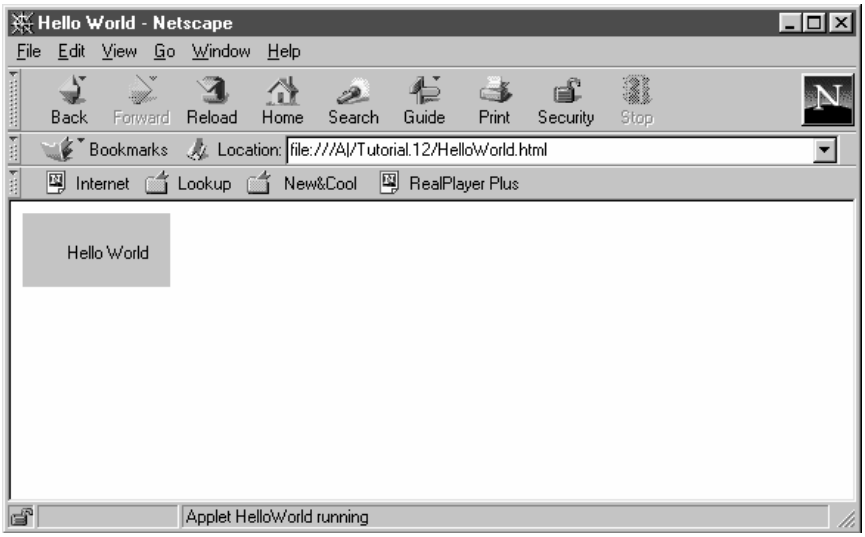


图 12-10　Navigator 中的 HelloWorld 小应用程序

提示：使用 JDK AppletViewer 命令能够快速地显示含有小应用程序的 HTML 文档。AppletViewer 命令用来在不使用 Web 浏览器的情况下快速地显示小应用程序。在 JDK 的 bin 目录下，能够发现 AppletViewer 程序文件：appletviewer.exe。AppletViewer 是一个命令行程序。请使用类似于 c:\jdk1.2.2 \bin \appletviewer HelloWorld.html（假定 HelloWorld.html 位于当前目录下并且 JDK 安装在 c 驱动器的 jdk1.2.2 文件夹内）命令来运行 AppletViewer。

下一步，创建用来显示 Random Number 小应用程序的 HTML 文档：

1．启动文本编辑器或 HTML 编辑器并创建新文档。

2．输入起始的<HTML>和<HEAD>节以及起始的<BODY>标签。

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

3．输入<APPLET CODE="RandomNumber.class" WIDTH=425 HEIGHT=50>创建包含 Random Applet 小应用程序的<APPLET>标签。

4．输入结束的</APPLET>、</BODY>和</HTML>标签。

```
</APPLET>
</BODY>
</HTML>
```

5．在 Data 磁盘的 Tutorial.12 文件夹内将文件存为 RandomNumber.html。在 Web 浏览器内打开此文件。尽管产生数可能不同，但是显示的结果应该与图 12-11 相似。单击浏览器的刷新或重新载入按钮来改变随机数。

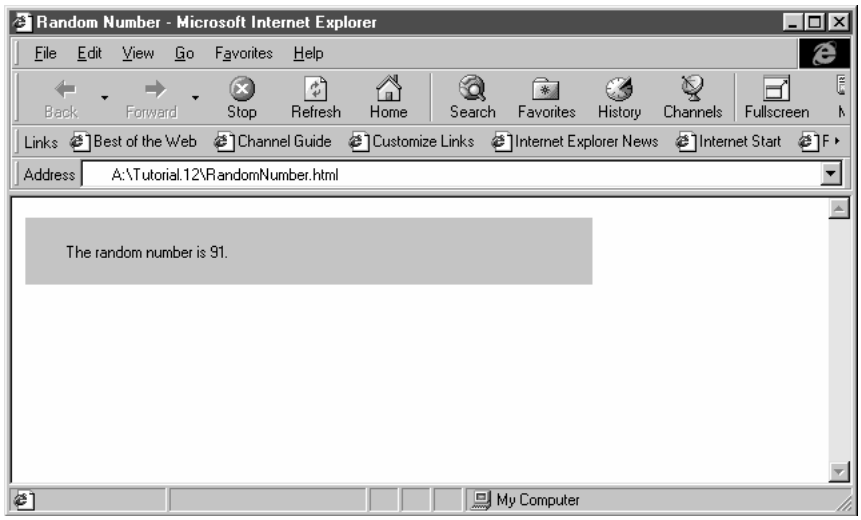


图 12-11　Web 浏览器内的 RandomNumber 小应用程序

6．关闭 Web 浏览器窗口。

下一步，将 Random Number 程序转换成猜数游戏。

12.1.9 使用 JavaScript 控制 Java Applet

对 Web 页而言，Java 小应用程序作为对象运行，类似于 JavaScript 对象，如 Window 对象和 Document 对象。以调用 JavaScript 对象的方法和属性相同的方式来调用小应用程序的方法和读写它的属性（只要它们是共有的）。使用 JavaScript 控制 Java 小应用程序最困难的任务是找出能够控制的共有方法和属性。若小应用程序不是编写的，发现此信息的惟一途径是参阅小应用程序附带的文档或询问小应用程序的开发人员。在此节，我们是编写自己的小应用程序并且知道能够使用的方法和属性。

有两种方法在 JavaScript 代码内引用小应用程序。能够使用在<APPLET>标签内所赋给的 NAME 属性或使用 applet[]数组内它的下标。可以将其中的一种小应用程序引用以及所想使用的方法或属性添加在 Document 对象的后面。applets[]数组以它们出现的顺序包含了一组 Web 页上的小应用程序。页面上的第一个小应用程序引用是 applets[0]，第二个小应用程序引用是 applets[1]，依此类推。引用的下标比它们出现的次序少 1（因为数组起始下标为 0）。还可用 NAME 属性来引用每个小应用程序。图 12-12 显示了一个简单的 HTML，它调用 AnimationSequence.class 小应用程序的 start()方法。文档使用<BODY>标签内的 onLoad 事件处理器来调用小应用程序的 start()方法。注意在<APPLET>标签内已将小应用程序的 NAME 属性赋为 animationApplet。

```
<HTML>
<HEAD><TITLE>Hello World</TITLE></HEAD>
<BODY onLoad="document.animationSequence.start();">
<APPLET CODE="AnimationSequence.class"
WIDTH=100 HEIGHT=100 NAME="animationApplet"></APPLET>
</BODY>
</HTML>
```

图 12-12 调用小应用程序的 start()方法的 HTML 文档

下一步，将为 RandomNumber.java 文件添加自定义的方法，它们为程序提供了全部功能。后面，将在 JavaScript 内将自定义的方法作为 RandomNumber 小应用程序的属性调用。

为 Random Number 程序添加产生随机数的代码：

1．返回到文本编辑器内的 RandomNumber.java 文件并立即在 Data 磁盘的 Tutorial.12 文件夹内将它存为 RandomNumberGame.java 文件。

2．将类首部内的类名改为 RandomNumberGame，那么首部应为 public class RandomNumberGame extends Applet {。

3．在 start()方法内，将 displayText = "The random number is " + randomNumber + ".";代码修改为 displayText = "Guess a number between 0 and 100.";。

4．在 paint()方法后，类反括弧前添加下面的 tooLow()和 tooHigh()方法，若用户猜错，

在 JavaScript 内将调用它们。此方法改变 displayText 变量并将 guessCount 变量增 1，接着调用 repaint()方法执行 paint()方法。

```
public void tooLow() {
        displayText = "Sorry! You guessed too low.";
        ++guessCount;
        repaint();
}
public void tooHigh() {
        displayText = "Sorry! You guessed too high.";
        ++guessCount;
        repaint();
}
```

5．下一步，添加下面的 rightNumber()方法，若用户猜对，那么将继续执行。此方法将修改 displayText 变量，它包含了来自 guessCount 变量所猜的次数和来自 randomNumber 变量的正确随机数。

```
public void rightNumber() {
        displayText = "You guessed correctly in " +
        guessCount
                + " tries! " + "The number is " +
                randomNumber + ".";
        repaint();
}
```

6．最后，添加最后一个方法 giveUp()，在用户单击 Web 页上的“ I Give Up, What Is It? ”按钮时调用它。

```
public void giveUp() {
        displayText = "The number is " + randomNumber
                + ". Better luck next time!";
        repaint();
}
```

7．保存和编译 RandomNumberGame.java 文件。

下一步，修改 RandomNumber.html 文件，以便包含控制 RandomNumber 小应用程序的方法和属性的 JavaScript 代码。

修改 RandomNumber.html 文件，包含控制 RandomNumber 小应用程序的方法和属性的 JavaScript 代码：

1．返回到文本编辑器或 HTML 编辑器内的 RandomNumber.html 文件，并立即在 Data

磁盘上的 Tutorial.12 文件夹内将它存为 RandomNumberGame.html。

2．将<APPLET>标签的 NAME 属性改为 RandomNumberGame.class 文件。接着在<APPLET>标签的反括弧前添加 NAME = "RandomNumberGame"为 RandomNumber 小应用程序创建一个名称。

3．在起始的<BODY>标签后添加<H2>Guessing Game</H2>。

4．在</HEAD>结束标签前为<SCRIPT>节添加起始的标签。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!-- HIDE FROM INCOMPATIBLE BROWSERS
```

5．按下回车键并为 JavaScript 函数 checkGuess()添加起始首部，它只传入一个 guess 参数：function checkGuess(guess) {。将使用一个表单按钮来执行 checkGuess()函数。

6．按下回车键并输入下面的代码，它创建一个 answer 变量并将 RandomNumberGame 对象（在 Document 对象的后面）的 randomNumber 属性赋给它。

```
var answer = document.RandomNumberGame.randomNumber;
```

7．在 answer 变量声明后，添加下面的 if…else 结构，它根据比较 guess 变量和 answer 变量所返回的结果来调用 RandomNumber 小应用程序不同的方法。

```
if (guess > answer)
        document.RandomNumberGame.tooHigh();
else if (guess < answer)
       document.RandomNumberGame.tooLow();
else
       document.RandomNumberGame.rightNumber();
```

8．为 checkGuess()函数添加反括弧以及输入结束的<SCRIPT>标签。

```
}
// STOP HIDING FROM INCOMPATIBLE BROWSERS -->
</SCRIPT>
```

9．最后，在结束的</BODY>标签前添加下面的表单和<H3>标签。在表单内，“ Guess ”按钮调用<SCRIPT>节内的 checkGuess()函数。请注意‘ I Give Up, What Is It? ’按钮的 onClick 事件处理器直接调用了 RandomNumberGame 小应用程序的 giveUp()方法。

```
<FORM NAME="guessForm">
<INPUT TYPE="text" NAME="guessField">
<INPUT TYPE="button" VALUE=" Guess " onClick=
```

```
        "checkGuess(document.guessForm.guessField.value);">
<INPUT TYPE="button" VALUE=" I Give Up, What Is It? "
        onClick="document.RandomNumberGame.giveUp();">
</FORM><P>
<H3>Click your Refresh or Reload button to start
over.</H3>
```

10．保存文件，接着在 Web 浏览器内打开它。输入一些数测试此程序。图 12-13 显示了程序首次打开时 Navigator 内程序显示的结果。

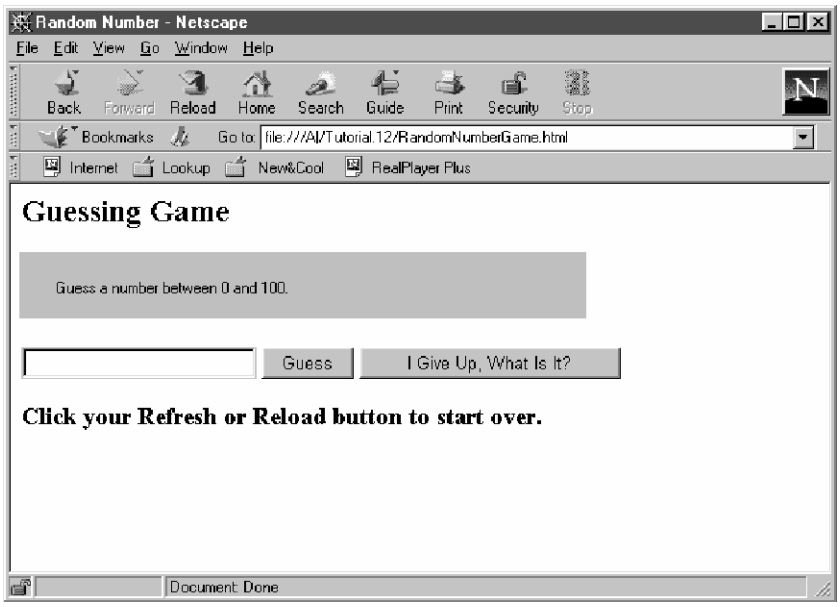


图 12-13　Navigator 中的 RandomNumberGame.html

11．关闭 Web 浏览器窗口。

12.1.10　总结

- ✧ Applet 是 Java 程序，它在 Web 页内运行。
- ✧ Navigator 使用特殊的程序扩展插件来显示嵌入式数据，而 Internet Explorer 使用 ActiveX 控件来显示嵌入式信息。
- ✧ 为了让 JavaScript、Java 和插件彼此之间相互通信，Navigator 使用了众人皆知的 LiveConnect 技术。Internet Explorer 使用 ActiveX 脚本来让 JavaScript 和 Java 以及 ActiveX 控件交互。
- ✧ 封装是指所有代码和所需的数据均包容在对象内。
- ✧ 围框是在 Web 页上小应用程序在其中执行的矩形区域。

- ✧ 面向对象编程是指创建可重用的软件对象，并且能够容易地与其他程序结合。
- ✧ 对象是能够作为独立单元或组件的编程代码和数据。
- ✧ Java 结构是中性的，意思是它能够在任何平台上运行。平台是指操作系统以及它的硬件类型。
- ✧ Java 虚拟机（Java VM）是 Java 编程语言的语言解释器。对每种平台，Java 编程语言均有不同的 Java VM。
- ✧ JavaScript 和 Java 具有多数相同的语法和语言结构，因此它们均从 C 编程语言继承而来。
- ✧ Java 开发包（JDK）是创建 Java 程序的基本编程环境，对几种平台（包括 Windows NT/95/98、Solaris 和 Macintosh）均有相应的 JDK。
- ✧ 在面向对象编程中，类是方法、属性和其他性质的结合。
- ✧ 面向对象语言（如 Java）根据类创建对象。
- ✧ 方法是一种类似于函数的结构，它含有类语句和过程。
- ✧ 为新 Java 类添加方法、属性和其他类来构建 Java 程序。
- ✧ public 访问修饰符决定了类访问许可的访问类型。
- ✧ 包或类库是相关类的集合。
- ✧ 在首次运行单机 Java 程序时自动调用 main()方法。
- ✧ public 关键字是方法描述符，它指定了方法在类外也能够被访问。
- ✧ static 关键字表示在计算机内存中仅存在一份方法拷贝，而不管类实例数目。
- ✧ void 关键字意味着方法将不返回值。与 JavaScript 方法不一样，Java 方法要么必须返回一个值，要么在方法首部包括 void 关键字。
- ✧ javac 程序是命令行程序，它将 Java 程序编译为具有.class 后缀的文件。
- ✧ 字节码是编译后的 Java 格式，它能够被 Java VM 执行。
- ✧ JDK\bin 目录下的 Java 程序是所在平台的 Java VM 并且能够被用来执行编译后的字节码。
- ✧ 小应用程序使用四个方法：init()、start()、stop()和 destroy()，它们在小应用程序的不同执行点自动运行。
- ✧ extends 关键字让新类基于另一个类的方法、属性和其他性质。
- ✧ paint()方法在浏览器窗口内显示小应用程序的可视化组件，并且在浏览器窗口最小化、最大化或缩放时自动运行。
- ✧ 使用 repaint()方法调用 paint()方法。
- ✧ 在 Java 内声明变量时，必须指明数据类型。
- ✧ 在类级别声明的变量被称为实例变量。实例变量也是 Java 内用来引用对象属性的术语。
- ✧ 类型转换将某种数据类型变量内的值拷贝到另一种数据类型的变量内。
- ✧ <APPLET>标签指明在 HTML 文档内包括小应用程序。使用四种属性来创建<APPLET>标签：CODE、WIDTH、HEIGHT 和 NAME。

✧　applets[]数组以它们出现的顺序包含了一组 Web 页上的小应用程序。

12.1.11　问题

1．在 Web 页内运行的 Java 程序被称为____。
a．插件
b．Active 控件
c．小应用程序(applet)
d．scriplet
2．____意味着所有的代码和所需的数据均包容在对象自身内。
a．多态
b．封装
c．黑箱实现
d．代码隐藏
3．____是 Web 页上小应用程序在其内执行的矩形区域。
a．围框
b．子文档
c．小应用程序容器
d．小应用程序帧
4．____指创建可重用的软件对象，并且能够容易地与其他程序结合。
a．过程化编程
b．面向对象编程
c．代码重用
d．开发结构
5．Java 代码能够在任何平台上运行是因为____。
a．程序员需要为每种平台编写不同的程序
b．在编译程序时自动创建不同版本的 Java 程序
c．Microsoft 设计它能够与所有操作系统协作
d．结构中性
6．____是 Java 编程语言的语言解释器。
a．Java RealPlayer
b．Sun 虚拟机
c．Java 虚拟机
d．通用 Java 编译器
7．____是创建 Java 程序的基本开发环境。
a．Java 开发包
b．Microsoft Applet Tool

c．Borland Cafe
d．ActiveX Control Pad
8．____是类似于函数的结构，它包含了类语句和过程。
a．过程
b．类容器
c．方法
d．函数源文件
9．Java 文件 PayrollApplet 类首部的正确语法是____。
a．public class PayrollApplet {
b．public PayrollApplet {
c．public class (PayrollApplet) {
d．public class Header {
10．相关类集合被称为包或____。
a．类容器
b．源对象
c．类结构
d．类库
11．____关键字使方法能够在其类外访问。
a．static
b．open
c．public
d．final
12．哪种 Java 方法与 JavaScript 的 write()和 writeln()方法相似？
a．screen()和 screenln()
b．print()和 println()
c．output()和 outputln()
d．writeTo()和 writeToln()
13．哪种程序将 Java 代码编译成具有.class 后缀的文件？
a．java
b．javac
c．javah
d．compile
14．能够被 Java 解释器执行的、编译后的 Java 代码被称为____。
a．字节码
b．机器码
c．源码
d．类码

15．作为单机 Java 程序的入口点方法是____方法。

a．start()

b．entry()

c．begin()

d．main()

16．下面哪种不是自动在小应用程序执行点运行的四种小应用程序方法?

a．load()

b．start()

c．stop()

d．destroy()

17．保存在 AccoutingApplet.java 文件内的小应用程序的正确的类首部是____。

a．public class AccountingApplet{

b．public class AccountingApplet extends Applet {

c．public class Applet AccountingApplet{

d．public class Applet extended AccountingApplet{

18．____方法在浏览器窗口内显示小应用程序的可视化组件，并且任何时候浏览器窗口最小化、最大化或缩放时自动执行。

a．paint()

b．draw()

c．display()

d．view

19．下面哪条语句用来在 Java 中声明字符串变量？

a．var String textString = "This is my text string";

b．String textString = new String("This is my text string");

c．var textString = new String("This is my text string");

d．textString = new String("This is my text string");

20．为了让实例变量能够在类外使用，它必须具有____访问修饰符。

a．static

b．final

c．open

d．public

21．Math 类的 random()方法返回介于____之间的数。

a．0.0 和 1.0

b．0.0 和 10.0

c．1.0 和 10.0

d．1.0 和 100.0

22．使用____标签将小应用程序添加到 HTML 文档中。

a．<CLASS>

b．<JAVA>

c．<OBJECT>

d．<APPLET>

12.1.12 练习

1．在 Data 磁盘的 Tutorial.12 文件夹内创建 Java 程序 PersonalInfo.java。在程序的 main()方法内，使用 print()和 println()方法将的姓名、地址和生日输出到屏幕上。

2．创建一个包含了存储时薪和工作小时数变量的 Java 程序。显示总薪水、代扣所得税（它是总薪水的 15%）以及净工资。在 Data 磁盘的 Tutorial.12 文件夹内将文件存为 Paycheck.java。

3．创建一个声明了用来存储房间长和宽英寸数变量的 Java 小应用程序。为变量使用合适的数据类型。将值赋给变量，接着计算房间地面空间的平方英寸数并将结果保存在另一个变量中。在 Data 磁盘的 Tutorial.12 文件夹内将程序存为 RoomSize.java。下一步，创建一个检索 RoomSize.java 文件的变量值的 HTML 文档。在 HTML 文档内，使用说明性的文本和变量显示房间尺寸和地面空间。例如，“ The room is 10 feet by 12 feet. ”。在 Data 磁盘的 Tutorial.12 文件夹内将 HTML 文档存为 RoomSize.html。

4．在 Data 磁盘的 Tutorial.12 文件夹内创建 Java 程序 FavoriteMovies.java。程序应该包含五种方法：start()方法、paint()方法、将三个所喜欢的戏剧女演员赋给实例变量的方法、将三个所喜欢的戏剧名赋给实例变量的方法以及将绝对喜欢的电影名赋给实例变量的方法。每种方法都应该调用 repaint()方法在小应用程序的围框内显示它所控制的变量的内容。下一步，创建一个包含了三个按钮（Favorite Comedies、Favorite Dramas 和 Favorite Movie）的 HTML 文档。每个按钮应该调用 Java 程序内合适的方法。在 Data 磁盘的 Tutorial.12 文件夹内将 HTML 文档存为 FavoriteMovies.html。

5．创建一个含有五个变量的小应用程序：employee、employer、jobTitle、startingSalary 和 age。其中 startingSalary 为长整型，age 为整型变量，其余的变量作为字符串对象。用一个图形对象的 paint()方法在小应用程序的边框中将变量显示出来。保存文件在文件夹 Tutorial.12 中，文件名为 JobInfo.java。用一个 HTML 表单收集这些信息，然后当用户按下一个按钮时发送这些信息，作为一个小应用程序中名为 reassignValues()的方法的参数。将 HTML 文档保存为 JobInfo.html，保存在文件夹 Tutorial.12 中。为了在 Java 方法的开始创建参数，必须说明每个参数的数据类型，该类型和上述变量的类型一致。注意，对于字符串参数，不用使用完整的字符串对象说明语句 String variable=new String();，简单地将参数说明为字符串类型即可。例如，要创建一个名为 myMethod()的方法，其中有一个字符串参数、一个长整型参数和一个双精度浮点类型参数，可以声明如下：public void myMethod(String stringArgument, long numberArgument, double floatArgument){。

6．创建一个小应用程序版的温度转换计算器，可以将华氏温度和摄氏温度相互转换。保存文件为 ConvertTemperature.java，在文件夹 Tutorial.12 中。将华氏转换为摄氏，需要从华氏温度减去 32，然后再乘以 0.55。将摄氏转换为华氏，需要乘以 1.8，然后加上 32。在一个 HTML 表单中收集要转换的数字，然后将其送到小应用程序中。用浮点数作为其中的参数和变量的数据类型。在小应用程序的边框中，显示转换后的温度。保存 HTML 文档为 ConvertTemperature.html 在文件夹 Tutorial.12 中。

12.2　LiveConnect、插件和 ActiveX

本节目标：

在本节里将学习：

- ✧ 关于 Java 包和 LiveConnect
- ✧ 关于在 Java 和 Netscape JavaScript 之间的数据转换
- ✧ 如何用 Java 控制 Netscape JavaScript
- ✧ 如何在 Netscape JavaScript 中直接存取 Java 类
- ✧ 如何操作嵌入数据（插件和 ActiveX 控件）

12.2.1　总览

在第 12.1 节里介绍了 Java 编程和如何用 JavaScript 控制 Java 小应用程序。本节里将学习如何用 Java 和 LiveConnect 在 Navigator 中操作 JavaScript 程序。还会学习在 Navigator 中，LiveConnect 是如何允许 JavaScript 程序直接存取 Java 类与插件通信。本节的主体不能应用于 IE，因为 IE 不支持 LiveConnect 和插件，也不允许 Java 程序操作 JavaScript。相比于 Netscape，IE 用 ActiveX 技术来显示和操作嵌入对象。ActiveX 技术将在本节最后论述。

12.2.2　Java 包和 LiveConnect

Java 语言用的包和类被安置在一系列文件夹中。最顶层的文件夹为每个 Java 包包含了相应的子文件夹。每个包的文件夹为包中的每个类包括了以.class 为后缀的类文件。Java 包和它们的类结构包含在一个压缩的 JAR 文件中，JAR 文件放置在 JDK 目录中。由于 Java 语言的不同特征，存在着各种不同的 JAR 文件。最常用的 JAR 文件是 rt.jar（rt 是 runtime 的缩写），在 jdk1.2.2 目录的 jre\lib 文件夹中。rt.jar 文件包括了最常用的 Java 包，包括 java.lang 和 java.awt 包。

任何时候，当编译一个 Java 程序时，Java 编译器打开 rt.jar 文件（和其他的程序需要

的 JAR 文件），并且解开程序需要的包和类。图 12-14 显示了包含在 rt.jar 中的部分文件夹的列表，还有一些包含在 java.awt 包中的类。

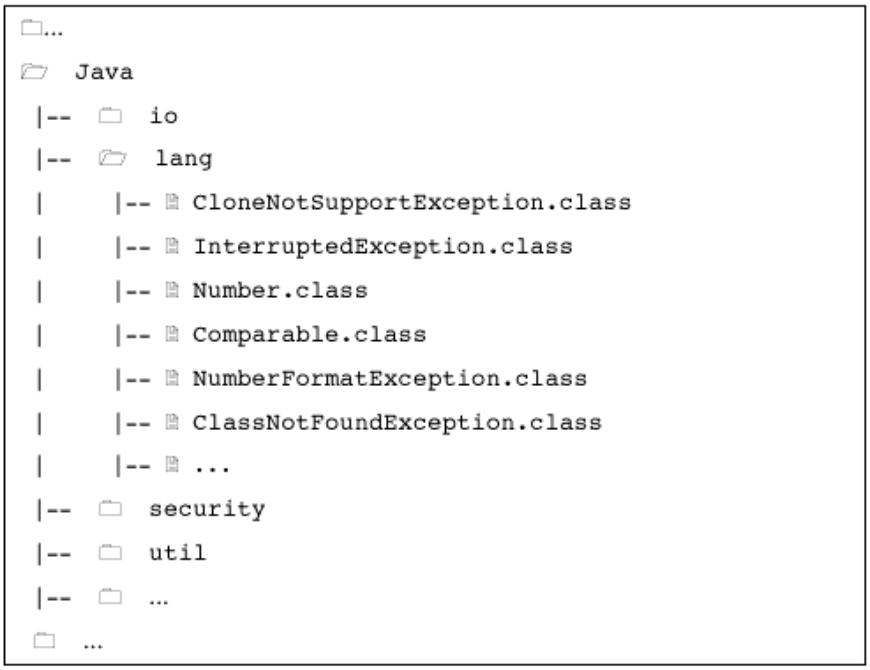


图 12-14　包和类结构

在 Navigator 中，LiveConnect 包使 JavaScript 能够和 Java 之间相互通信。LiveConnect 包提供了 Navigator 存取核心 Java 的功能，并且包含了 Java 程序控制 JavaScript 的必需的类。Java 知道在\lib 目录寻找 classes.zip 文件，但是 Java 并不会自动地知道到哪里找 LiveConnect 包。LiveConnect 包包含在压缩文件 java40.jar 中，该文件在计算机的 Netscape 文件夹中。

提示：JAR 文件和 ZIP 文件一样，是一种压缩了的文件。

提示：要在 Navigator 中用 LiveConnect，必须在优先对话框的高级页框中，选中 Java 和 JavaScript 选项。

java40.jar 的定位依赖于使用的平台，还有是否在使用 Communicator 或者 Navigator。为了让 Java 能够定位 LiveConnect 包，必须定位 java40.jar 文件所在目录，并且将 CLASSPATH 环境变量指向该目录，就像指向 JDK/bin 目录一样。CLASSPATH 环境变量告诉 Java 虚拟机和 JDK 应用到哪里去找 Java 应用程序需要的类。设置 CLASSPATH 环境变量的方法根据平台而不同。在 Windows NT 中，可以用控制面板中的系统图标来设置环境变量。在系统属性对话框中，单击环境页框，然后添加如下的环境变量。输入的路径取决于 JDK 的安装路径和系统中 java40.jar 文件的位置。

```
CLASSPATH=C:\Program Files \Netscape \Navigator \Program \Java \
Classes \java40.jar
```

提示：Navigator 是 Netscape 的 Web 浏览器产品。Communicator 是 Netscape 全套的 Internet 工具包，包括 Navigator、Messenger、Calendar、Composer 和 AOL Instant Messenger。

如果在 Windows 95/98 系统上工作，那么需要在 autoexec.bat 文件中添加一个 SET 语句来创建 CLASSPATH 环境变量。例如，添加如下的 SET 语句到 autoexec.bat 中，用来在 Windows 95/98 系统中创建 CLASSPATH 环境变量。同样地，输入的路径取决于 JDK 的安装路径和系统中 java40.jar 文件的位置。

```
SET CLASSPATH=C:\Program Files \Netscape \Navigator \Program \
Java \Classes \java40.jar
```

需要注意的是，必须在一行里输入 SET 语句，而不能和上例一样分成两行。在编辑完 autoexec.bat 文件后，需要重新启动系统以使 CLASSPATH 环境变量产生效用。

提示：如果使用 Windows 以外的平台，查看操作系统的帮助文档，看应该如何设置 CLASSPATH 环境变量。此外，如果使用 Sun JDK 以外的 Java 开发程序，请检索开发文档，查看应该如何设置 CLASSPATH 环境变量。

12.2.3　Java 和 JavaScript 之间的数据转换

当用 JavaScript 控制 Java 程序，或者 Java 控制 JavaScript 程序时，通常在两者之间需要进行表单数据的传送。通常这种传送会引起问题，因为 Java 和 JavaScript 的数据类型并不匹配。例如，JavaScript 仅仅有两种数字类型，整型和浮点数字类型，而 Java 有六种类型。另外，因为 Java 是一种很强的类型语言，必须从 JavaScript 往 Java 变量传递正确的数据类型，否则会产生错误。例如，不能往一个 Java 整型变量传递一个 JavaScript 的字符串变量。传递正确的数据类型可能让 JavaScript 程序员产生困惑，因为在程序执行时 JavaScript 变量可能会发生改变。但是，Java 数据类型却不在执行时发生变化。要想让 Java 和 JavaScript 程序一起工作，需要知道在这两种语言之间数据类型是如何转换的。表 12-2 列出了在两者之间，数据类型是如何转换的。

表 12-2　在 JavaScript 和 Java 之间的数据类型转换

JavaScript 数据类型	Java 数据类型
array	Array 对象
boolean	boolean
floating-point 或 integer	byte、char、short、int、long、float 和 double
null	null
undefined	void

续表

JavaScript 数据类型	Java 数据类型
所有其他 JavaScript 对象	JSObject 封装器
JavaObject 封装器	所有其他 Java 对象

从 JavaScript 传递数据到 Java 时要特别小心，特别是在传递数字数据类型时。从 JavaScript 传递到 Java 的数字类型变量转换为最匹配的 Java 数字数据类型。但是，如果 Java 程序需要一个浮点类型变量，而从 JavaScript 传递来的变量转换成了整型变量的话，就会产生一个错误。通过将传入的类型转换为正确的数据类型，就可以处理这种状况。在第 12.1 节里，已经学习了类型转换。

从 Java 传递数据到 JavaScript 不像想象中的那么复杂，因为 JavaScript 是一种松散的类型语言。如果 JavaScript 接收了 Java 六种数字类型中的任意一种，它们将被转换成浮点或整型变量。

从 JavaScript 传递到 Java 的复杂对象，被封装在 JSObject 封装器中，从 Java 传递 JavaScript 的复杂对象被封装在 JavaObject 封装器中。封装器是一个类或者对象，封装了其他的相关信息。因为在 Java 和 JavaScript 之间传递的复杂对象（例如 JavaScript 历史对象），不能在对方环境里被转化为一个等同的对象，所以将这些复杂对象封装起来，以便对方能够识别。然后对方语言可以操作封装器对象，并且从中提取信息。简单一些的对象，例如数字和布尔类型，也被封装在 JSObject 类中，但是可以被转换为一个对应的 Java 数据类型。在本节的后面，将会用 JSObject 类进行工作。

12.2.4 用 Java 控制 JavaScript

可以用 JavaScript 控制 Java 语言，当然也可以用 Java 控制 JavaScript 语言。但是如果设计一个控制 JavaScript 的小应用程序，重要的是要严格地控制程序的可移植性，因为小应用程序需要寻找特定名称的 JavaScript 函数和 Web 页面中的变量。可能有很好的理由创建该种类型的小应用程序。用一个先前的例子，创建一个收集财务信息的表单，该表单提交数据到一个 CGI 程序。因为该表单包含几个页面，会发现，与其开发一个完整的 Java 应用程序，不如用 HTML 创建小程序并提交到 CGI 容易的多。但是，还需要利用 Java 中的一些计算方法，在 JavaScript 中没有这些方法。在这种情况下，需要一个 Java 的小应用程序从表单收集数据，进行计算，然后将计算结果传送给表单。还可以用 applet 执行表单的提交事件。在这种特定情况下，小应用程序就没必要考虑可移植性问题，因为它和 Web 页面结合的非常紧密。

要想允许 Java 小应用程序控制 JavaScript，必须做到：

- ✧ 在 applet 中引入 JSObject 和 JSException 类。
- ✧ 用 JSObject 类的方法存取包含小应用程序的浏览器窗口和 JavaScript 的变量和函数。

✧　在<APPLET>标签中添加 MAYSCRIPT 属性。

为了了解这三个步骤，我们来对第 12.1 节中创建的 RandomNumberGame 小应用程序进行改造，让其能够控制 HTML 文档，存取 JavaScript 对象、变量和方法。在原来的程序中，用一个 JavaScript 函数计算用户的猜测，并且调用 applet 中的方法，将文本写到 applet 的弹出对话框中，输出的文本取决于用户的猜测。在新程序里，applet 通过从表单文本域（名为 guessField）直接取值来计算用户的猜测。然后小应用程序调用 JavaScript 的 alert() 方法来通知用户他们的猜测是否是正确的。在下节要练习的主要目的是显示一个 applet 如何读取 JavaScript 对象和变量，以及执行 JavaScript 方法。

引入 JSObject 和 JSException 类

要用一个小应用程序控制 JavaScript，首先必须输入包含在 LiveConnect 包中的两个类：JSObject 和 JSException。JSObject 类包含了允许 Java 和 JavaScript 之间交互的方法，并且可以作为一个 JavaScript 对象封装器进行工作。JSException 类将 JavaScript 的错误传递回 Java 类。这两个类都包含在 LiveConnect netscape.javaScript 包中。要将这两个类引入一个 applet，需要在程序的开始包括语句 import netscape. javascript.*;。当使用该语句时，注意一些系统经验编译错误。如果在引入 netscape.javascript 包时，收到一个编译错误，尝试着将 JSObject 和 JSException 类分别引入：

```
import netscape.javascript.JSObject;
import netscape.javascript.JSException;
```

下面，引入 LiveConnect 包到 RandomNumberGame 小应用程序中：

1．在文本或 HTML 编辑器中打开文件 RandomNumberGame.java，并将其另存为 RandomNumberGame2.java 文件，保存在文件夹 Tutorial.12 中。

2．在引入 java.applet 类的语句前面，添加 import netscape.javascript.*;语句。

3．将类名称变更为 RandomNumberGame2，添加如下的语句 public class Random-NumberGame2 extends Applet{。

4．保存并编译文件。

帮助：如果文件不能正确编译，试着将引入 LiveConnect 包的单行语句转为两个分开的行:

```
import netscape.javascript.JSObject;
import netscape.javascript.JSException;
```

使用 JSObject 类方法

JSObject 类包含了用来存取和操作 JavaScript 的方法。表 12-3 列出了 JSObject 类的方法。

表 12-3 JSObject 类的方法

方 法	说 明
call(String functionName, Object args[])	执行一个 JavaScript 方法或函数
eval(String expression)	执行一个 JavaScript 字符表达式
getMember(String propertyName)	返回一个 JavaScript 对象属性的引用
getSlot(int index)	返回在一个 JavaScript 对象中的数组元素
getWindow(applet)	获得包含了 applet 的窗口的句柄
removeMember(String propertyName)	删除 JavaScript 对象的一个属性
setMember(String propertyName, Object value)	设置 JavaScript 对象的一个属性
setSlot(int index, Object value)	设置在 JavaScript 对象中的数组元素
toString()	返回 JavaScript 对象的字符取值

要用 JSObject 方法存取 JavaScript 程序，首先必须用 getWindows()方法来存取浏览器窗口，操作是通过窗口句柄进行的。一个句柄确定一个操作系统资源。在这种情况下，资源指的是浏览器窗口。可以将窗口句柄看作对 JavaScript 中 window 或 self 对象的引用。获取一个句柄的语句是 JSObject variableName =JSObject.getWindow(this);。语句中的 this 代表包含了该 applet 的当前窗口。下面的语句创建了一个名为 jsWindow 的句柄：

```
JSObject jsWindow =JSObject.getWindow(this);
```

在创建了一个句柄以后，需要在句柄后添加 getMember()方法，以返回对 JavaScript 程序中其他对象的引用。在 JavaScript 程序中对每个对象的引用都必须嵌套在 JSObject 类型后面。将引号引起来的对象名称作为 getMember()方法的参数。举例来说，要返回对文档对象的引用，并将其转换成 JSObject 类型。语句应为 JSObject javaScriptDoc=(JSObject) jsWindow.getMember("document");（假设已经创建了名为 jsWindow 的句柄）。下面的代码创建了一个 Web 浏览器的句柄，然后用 3 个 getMember()方法的实例存取表单中的文本输入域。

```
JSObject jsWindow =
        JSObject.getWindow(this);
JSObject jsDoc =
        (JSObject) jsWindow.getMember("document");
JSObject jsForm =
        (JSObject) jsDoc.getMember("ProductInfo");
JSObject jsElement =
        (JSObject) jsForm.getMember("CustomerName");
```

要注意，在上面的例子中，每个对象都被转换成 JSObject 类型。这种转换为每个对象创建一个 JSObject 封装器。在此时，对象仍然被封装在 JavaScript 中。要使用 Java 中一个对象的属性，必须将值转换到一个 Java 变量。当转换一个值到 JSObject 封装器时，数据转

换规则就会起作用。举例来说，上例中的 jsElement 对象是 JavaScript 表单的文本输入域。要将文本域的值转换为一个名为 javaCustomerName 的 Java 变量，需要用一个包含 getMember()方法的语句返回取值，然后将该值转换到字符串对象。在下面的代码中，getMember()方法的参数是用引号引起来的该表单元素的 value 属性。

```
String javaCustomerName =
(String) jsElement.getMember("value");
```

下面将往 RandomNumberGame2.java 中添加一个 checkGuess()方法，该方法类似于先前在 JavaScript 中创建的 checkGuess()函数。在 checkGuess()方法中，将创建引用 JavaScript 文档对象、guessForm 表单和 guessField 文本域的对象。

往 RandomNumberGame2.java 中添加一个 checkGuess()方法：

1．在文本或 HTML 编辑器中打开文件 RandomNumberGame2.java。

2．在 start()方法的前面，添加 checkGuess()方法的开头部分：

```
public void checkGuess() {
```

3. 紧跟着 checkGuess()方法的开头，添加下面的语句，用来创建引用 JavaScript 文档对象、guessForm 表单和 guessField 文本域的对象。

```
JSObject gameWindow = JSObject.getWindow(this);
JSObject gameDoc = (JSObject)
        gameWindow.getMember("document");
JSObject gameForm = (JSObject)
        gameDoc.getMember("guessForm");
JSObject gameElement = (JSObject)
        gameForm.getMember("guessField");
String guessString = (String)
        gameElement.getMember("value");
```

4. 从 guessField 文本域返回的值转换到一个名为 guessString 的字符串变量。要比较包含在 guessString 和 randomNumber 变量中的数字，必须将字符串变量转换为整数，因为 randomNumber 变量是整型的。字符串转换为整型，需要将字符串变量传递到整数类的 parseInt()方法中，然后将结果赋给一个新的整型变量，代码如下：

```
int guess = Integer.parseInt(guessString);
```

5. 为 checkGuess()方法添加结束括号}。

6. 保存文件。

用到的另一个 JSObject 类的方法是 eval()方法，该方法在 Java 中执行 JavaScript 语句。

添加 eval()方法到一个句柄，并且将一个包含语句的字符串传递给它。例如，如果有一个名为 jsWindow 的句柄，则下面的 Java 代码弹出一个警告对话框，显示文本 Hello World。注意，传送到 alert()方法中的文本参数是用单引号引起来的。

```
jsWindow.eval("alert('Hello World');");
```

提示：Java 中的 eval()方法和 JavaScript 中的 eval()方法执行同样的功能。

下面，将往 RandomNumberGame2.java 中添加一个 if…else 结构，来比较 guess 变量和 randomNumber 变量，然后在 JavaScript 警告对话框中显示相应的消息。

往 RandomNumberGame2.java 中添加一个 if…else 结构：

1．在文本或 HTML 编辑器中打开文件 RandomNumberGame2.java。

2．在 checkGuess()方法的结束括号前，添加下面的 if…else 结构，来比较 guess 变量和 randomNumber 变量，然后根据比较结果，在 JavaScript 警告对话框中显示相应的消息。如果猜测的过低或过高，guessCount 变量就增加 1。

```
if (guess > randomNumber) {
        gameWindow.eval("alert('Sorry! You guessed too
high.')");
        ++guessCount;
}
else if (guess < randomNumber) {
        gameWindow.eval("alert('Sorry! You guessed too
        low.')");
        ++guessCount;
}
else {
        String rightGuess = new String();
        rightGuess = "alert('You guessed correctly in "
                + guessCount + " tries!"
                + " The number is " + randomNumber + ".')";
        gameWindow.eval(rightGuess);
}
```

3．最后，删除 tooLow()、tooHigh()和 rightNumber()方法。这些方法已经不再需要了，因为 checkGuess()方法已经替代了它们的功能。

4．保存并编译文件。在测试程序前，必须对包含它的 HTML 文档做些修改。

MAYSCRIPT 属性

在一个 Java 小应用程序中控制 JavaScript 的最后一个必须的步骤是，为包含 applet 的<APPLET>标签添加 MAYSCRIPT 属性。该属性给了一个 applet 在 Web 页面中操作 Java-

Script 代码的权限。例如，要允许一个名为 MyApplet.class 的小应用程序控制 JavaScript 代码，可以用语句<APPLET CODE="MyApplet.class" MAYSCRIPT>。

下面，将修改 RandomNumberGame.html 文件，以便使 RandomNumberGame2 小应用程序正确工作。

修改 RandomNumberGame.html 文件：

1．在文本或 HTML 编辑器中打开 RandomNumberGame.html 文件，并另存为 RandomNumberGame2.html 文件，保存在文件夹 Tutorial.12 中。

2．在文档的开头删除<SCRIPT>部分。由于用 Java 小应用程序来控制，JavaScript 的代码就不再需要了。

3．改变<APPLET>标签的 CODE 属性为 RandomNumberGame2.class。

4．在<APPLET>标签的结束括号前，添加 MAYSCRIPT 属性。

5．将在 Guess 按钮的 onClick 事件处理器中执行的代码改为 document.RandomNumberGame.checkGuess();，该语句调用 checkGuess()方法，以替代原来在<SCRIPT>部分中调用的 checkGuess()函数。

6．保存文件，在 Navigator 中打开它。输入一些猜测的数字测试程序。图 12-15 显示了程序在 Navigator 中的输出。

7．关闭 Web 浏览器窗口。

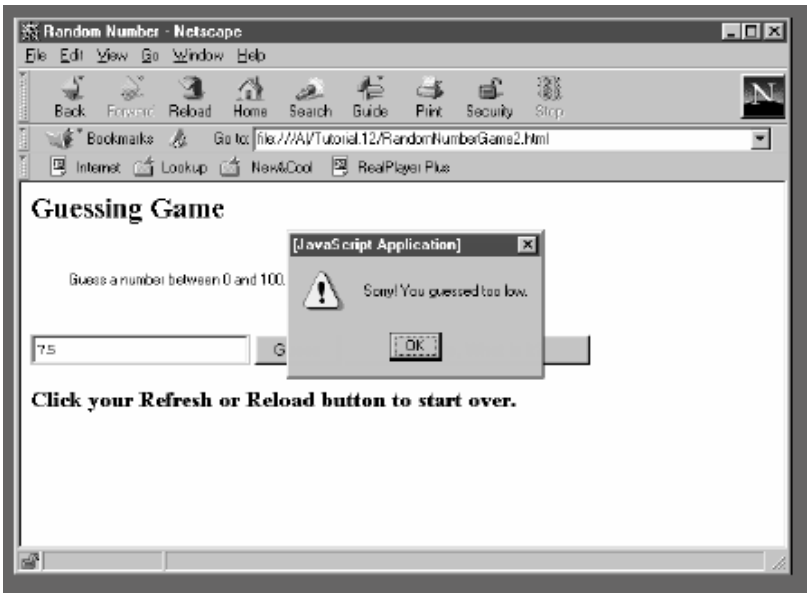


图 12-15　RandomNumberGame2.html 在 Navigator 中的输出

12.2.5　在 JavaScript 中直接存取 Java 类

在 LiveConnect 包中的 Java 包和类是 JavaScript 包对象的组成部分。Packages 对象提供

了 JavaScript 存取在 LiveConnect 中的 Java 包、类和方法的功能。利用对 Java 类的直接存取，可以不用创建 applet 就能增强 JavaScript 程序。举一个简单的例子，java.lang.System 类中的 getProperty()方法就返回了当前系统的信息。虽然很多 getProperty()方法返回的信息可以用 JavaScript 中的 Navigator 对象获得，但是，有一些信息却不能用 JavaScript 获得。用 getProperty()方法可以了解一个客户端计算机中，操作系统的体系结构和版本号，这两条信息就无法从 JavaScript 中的 Navigator 对象获得。

getProperty()方法只不过是为什么要在 JavaScript 中使用 Java 包的一个小例子。还可以执行更复杂的任务，比如用 java.awt 包来创建用户界面或创建新的设计元素。虽然在 Java 小应用程序中，执行这些任务可能更容易，但是在 JavaScript 中也是行得通的。在使用 Java 代码之前，需要注意的是，大部分常用的方法在 JavaScript 语言中已经实现了。例如，虽然 Java 包含了更多的操作字符串的方法，但是，更常用的是 JavaScript 中定义的较少的字符串方法。

提示：在 JavaScript 中使用 Java 类有许多限制，例如不能创建新类。并且 Java 代码受同样的安全规则的限制。

和其他的 JavaScript 对象不同，当使用 Packages 对象时，必须对字母 P 大写，以便和 JavaScript 中的关键字 package 区分开来。但是，当引用在 java、sun 和 netscape 包中的 Packages 对象时，就没有必要明确指出。下面的两个 getProperty()方法的调用，返回的操作系统的体系结构是一样的：

```
Packages.java.lang.System.getProperty("os.arch");
java.lang.System.getProperty("os.arch");
```

提示：在 JavaScript 中，保留关键字 package 用来解决将来的 JavaScript 中包的不兼容问题。

下面，修改在第 12.1 节中创建的 RandomNumberGame.html 文件，让文件可以直接存取 Java 类。

修改在第 12.1 节中创建的 RandomNumberGame.html 文件：

1．在文本或 HTML 编辑器中打开 RandomNumberGame.html 文件，文件存在 Tutorial.12 文件夹中，并另存为 RandomNumberGameDirect.html 文件。

2．在 checkGuess()函数开头前，添加下面的三行语句。第一条语句从 Java Math.random()方法返回一个数字，将其乘以 100，然后将结果赋给 answer 变量。第二个语句用 Java Math.round()方法将 answer 变量转为一个完整数字。第三个语句声明一个全局的变量 guessCount，来跟踪用户已经猜测的次数。

```
var answer = Packages.java.lang.Math.random() * 100;
answer = Packages.java.lang.Math.round(answer);
var guessCount = 0;
```

3．从函数 checkGuess()中删除语句 var answer =document.RandomNumberGame.randomNumber;。

4．将 checkGuess()函数中的 if…else 结构用下面的 if…else 结构代替：

```
if (guess > answer) {
        alert("Sorry! You guessed too high.");
        ++guessCount;
}
else if (guess < answer) {
        alert("Sorry! You guessed too low.");
        ++guessCount;
}
else {
        alert("You guessed correctly in " + guessCount
                + " tries!" + " The number is "
                + answer + ".");
}
```

5．删除<APPLET>…</APPLET>标签对。

6．修改 onClick 事件处理器代码，onClick="alert('The number is ' + answer + '.Better luck next time!');"。

7．保存文件，然后在 Navigator 中打开。文件应该和本章中的另两个版本的猜数程序有一样的功能。

8．关闭 Web 浏览器窗口。

12.2.6　嵌入数据

本章余下部分的目标是用 JavaScript 来操作嵌入数据。LiveConnect 为 JavaScript 提供了在 Navigator 中操作插件的能力。IE 提供了对插件的支持，但是在 IE 中通常用 ActiveX 替代 LiveConnect。

要学习在 JavaScript 中如何操作插件和 ActiveX 控件，可能用到文件夹 Tutorial.12 中的 hellcat.rpm 文件和 RealPlayer 程序。如果没有准备好，从 http://www.real.com/products/index.html 下载 ReaPlayer 安装程序，运行安装程序，将应用程序和插件及 ActiveX 控件一起安装。

插件

Web 浏览器显示两种基本的媒体类型：HTML 文档中的文本和图像，诸如.jpg 和.gif 文件，在 HTML 文件中由<IMG>标签引用。由于 Web 的飞速增长，在 Web 文档中包括其他类型媒体的要求也越来越强。现在，许多不同类型的媒体都是可用的，从 Word 文档到

其他静态信息，到声音、视频和动画。每种媒体类型都有不同的文件格式。此外，新的文件格式正在不断地开发出来。Web 浏览器不可能支持所有能够想象到的文件格式类型。

Netscape 解决了这个问题，通过助手应用程序打开 Navigator 不支持的文件类型。助手应用程序是一个用来打开 Navigator 不支持的文件类型的外部程序。例如，可以配置 Navigator，指定 Word 作为助手应用程序以打开.doc 文件。

助手应用程序的一个缺点是，打开的媒体不能作为 Web 页面的一部分显示，只是显示在一个单独的窗口中。在一个单独的窗口中打开媒体，适合某种类型的信息，例如字处理文档或电子制表软件。但是 Web 浏览器的设计者倾向于将其他一些类型的信息显示在浏览器的页面中，例如视频，作为页面的一个部分。要在一个页面中显示和执行另外类型的媒体，Netscape 介绍了插件技术。插件技术是由第三方开发者开发的软件组件，组件允许在 Navigator 窗口中显示和执行不同类型的信息。插件技术使 Netscape 避免为了支持新开发的文件类型不断升级 Navigator 的必要。

提示：在本章中用到的 RealPlayer 程序可以是一个应用程序或插件。

Navigator 4 本身包含了几个内置的插件。其中三种最常用的插件列在表 12-4 中。

表 12-4 Navigator 4 常用的内置插件

插　　件	说　　明
LiveAudio	播放 AIFF、AU、MIDI 和 WAV 声音文件
LiveVideo	播放 AVI 视频文件
QuickTime	播放 QuickTime 视频文件

提示：Netscape 保留了一个含有许多插件的描述和链接的页面，该页面可以用在 Navigator 中。地址为 http://home.netscape.com/plugins/index.html。

提示：插件是与平台有关的——只能使用支持平台的插件。

MIME 类型和文件扩展

回忆一下，MIME 类型是在 Internet 上交换文件的标准方法。当存取在服务器上的一个 Web 页面时，Navigator 启动一个插件或助手应用程序，根据页面的 MIME 类型，用指定的媒体类型来打开它。当一个插件第一次被安装时，自动地用 Navigator 注册 MIME 类型以及该类型文件的扩展名。偶尔，插件会用另一个不同的插件或助手程序来注册一个 MIME 类型和该类型文件的扩展名。举例来说，可能会下载一个插件，自动地配置用 Navigator 打开后缀为.avi 的视频文件。但是，可能倾向于用 Netscape 的 LiveVideo 插件打开后缀为.avi 的文件。可以配置 Navigator 4 来管理 MIME 类型和文件扩展名，通过在 Navigator 浏览器中 Preferences 对话框中的 Application 选项来配置。图 12-16 显示了在 Navigator 4 中配置 Preferences 对话框中的 Application 选项的例子。

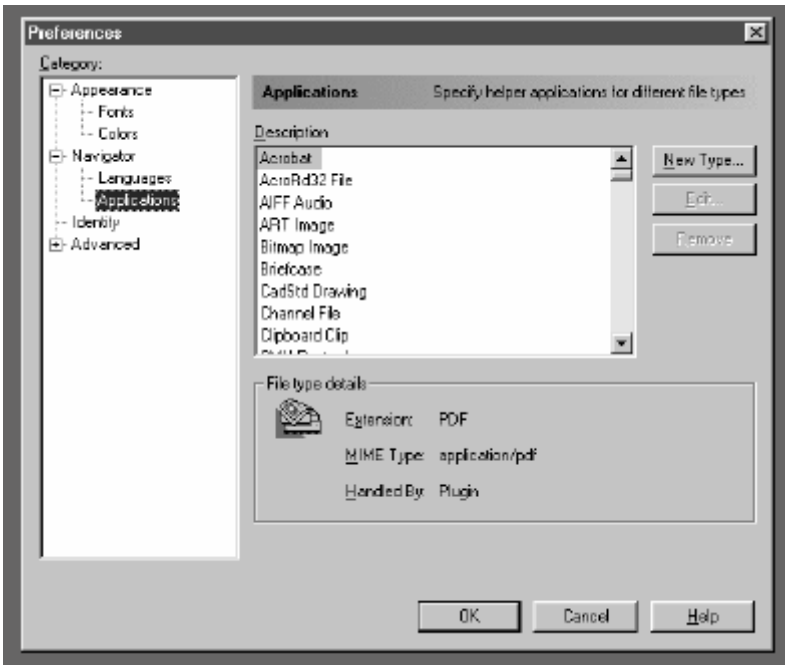


图 12-16　在 Navigator 4 中配置 Preferences 对话框中的 Application 选项

<EMBED>标签

当安装了一个插件或助手应用程序后，就可以通过添加<EMBED>标签，在 Web 页面中添加嵌入数据。<EMBED>标签在 Web 页面中创建一块矩形区域作为插件区域，插件区域和 applet 区域相似。表 12-5 列出了<EMBED>标签在 Navigator 和 IE 中有用的属性。

表 12-5　<EMBED>标签在 Navigator 和 IE 中有用的属性

属　性	说　明
HEIGHT	插件区域的高度
HIDDEN	决定插件是隐藏还是可见，默认为可见
NAME	与<EMBED>标记关联的一个名称
PALETTE	插件的调色板
SRC	在嵌入数据中显示的文档
UNITS	与高度和宽度属性有关的计数单位类型
WIDTH	插件区域的宽度

与大多数的 HTML 标签不同，<EMBED>标签可以包括一个插件特有的用户属性。举例来说，Netscape 的 LiveAudio 插件包括一个 AUTOSTART="true"属性，该属性在 Web 页面第一次载入时自动地播放一个音乐文件。

提示：要了解插件的属性，可以查阅插件的文档。

图 12-17 显示了一个 HTML 文档的例子，文档包含了一个.wav 后缀的嵌入的声音数据。文档中的<EMBED>标签包括了一个 AUTOSTART 属性，还有一个 HIDDEN 属性用来隐藏<EMBED>标签，因为在播放一个声音文件时，没有必要将其置为可见。

```
<HTML>
<HEAD><TITLE>LiveAudio Demo</TITLE></HEAD>
<BODY>
<EMBED SRC="sound_file.wav" NAME="audio" AUTOSTART="true" HIDDEN="true">
</BODY>
</HTML>
```

图 12-17　用<EMBED>标签播放声音文件

下面，将创建一个 HTML 文档，该文档用 RealPlayer 插件来播放 F6F“悍妇”飞机的视频。用 RealPlayer 的客户属性来配置<EMBED>标签。表 12-6 中列出了<EMBED>标签的客户属性。

表 12-6　RealPlayer 插件<EMBED>标签的属性

属　　性	说　　明
CONTROLS	设置显示的 RealPlayer 组件。有效的值包括 ALL、ControlPanel、InfoVolumePanel、ImageWindow、InfoPanel、StatusBar、PlayButton、StopButton、VolumeSlider、PositionSlider、PositionField 和 StatusField
CONSOLE	指定一个名称，用来管理 RealPlayer 的多个实例
AUTOSTART	当该属性设置为真时，就可以在 Web 页面第一次载入时，自动地执行一个 RealPlayer 插件
NOLABELS	关闭控制窗口中的标题，作者和版权标签文本，但是显示每个域的文本字符串

用 RealPlayer 插件来播放 F6F“悍妇”飞机的视频：

1．在文本或 HTML 编辑器中创建一个新的文档。

2．输入文档的开始标签。

```
<HTML>
<HEAD>
<TITLE>F6F Hellcat</TITLE>
</HEAD>
<BODY>
```

3．然后，在文档的开头添加<H2>Flight Characteristics of the F6F "Hellcat"</H2>。

4．输入下面的<EMBED>标签，该标签执行 RealPlayer 插件。标签中包括了 NAME 属性为 hellcat，该属性将在后面的 JavaScript 代码中用到。标签中还包含 AUTOSTART ="true"

属性，保证当 Web 页面载入时插件自动执行。将一个 ImageWindow 的值赋给 CONTROLS 属性，该属性创建一个单独的没有控制的视频窗口。

```
<EMBED NAME="hellcat" SRC="hellcat.rpm" AUTOSTART="true"
    WIDTH=220 HEIGHT=180 CONTROLS="ImageWindow">
```

5．添加下面的关闭标签：

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

6. 保存文件在 Tutorial.12 文件夹中，文件名为 RealPlayerPlugin.html。在 Web 浏览器中打开文件。视频应当在页面载入时播放。图 12-18 显示了在 Navigator 中页面的输出。

7. 关闭浏览器窗口。

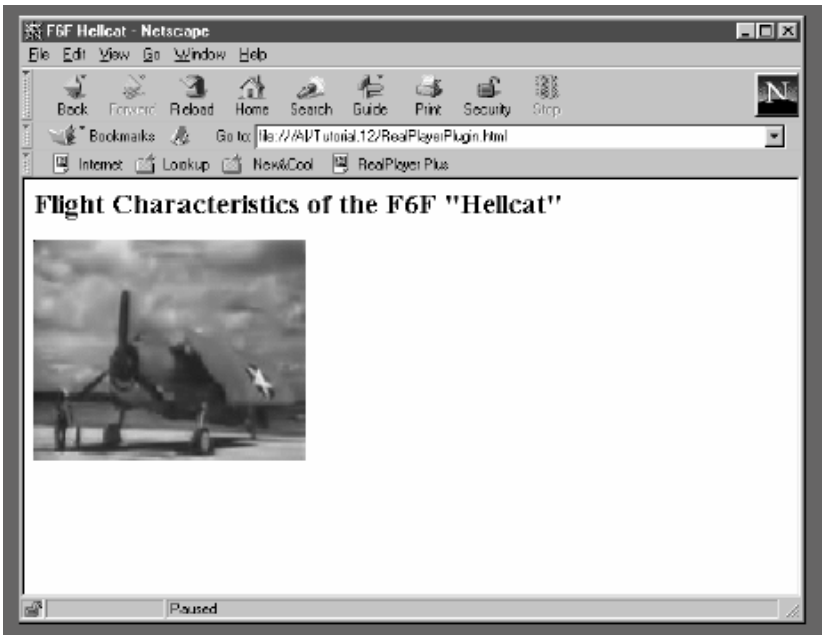


图 12-18　RealPlayerPlugin.html 在 Navigator 中的输出

下面，学习如何用 JavaScript 启动和停止插件的执行。

用 JavaScript 操作插件

在 JavaScript 代码中引用嵌入数据有两种方法：一是通过在<EMBED>标签中指定的 NAME 属性，二是用 embeds[]数组来引用。embeds[]数组包含了页面中嵌入数据的列表，按照嵌入数据的排列顺序为索引。页面中的第一个嵌入数据为 embeds[0]，第二个嵌入数据为 embeds[1]，依次类推。可以在文档对象中引用嵌入数据的数组或 NAME 属性，后跟需

要的方法或属性。可能注意到了，引用嵌入对象和 applets 的引用十分相似。就像在用 applets 时采用的方法，推荐用 NAME 属性引用嵌入数据，因为这种方法比较容易记忆。

通过存取插件的方法和属性，可以控制一个插件。这和控制小应用程序的方法是一样的。在文档对象中添加一个插件的 NAME 属性，后跟一个方法或属性名称。要在 JavaScript 中控制一个插件，必须首先了解该插件的属性和方法。可以查阅第三方开发者的文档来学习这些方法。大多数的插件开发者都提供包含了详细的技术信息的 Web 页面可供查询。例如，Netscape 的 LiveAudio 插件包含了各种控制声音文件播放的方法。图 12-19 中的 HTML 文档，用表单按钮和 onClick 事件处理器调用若干个 LiveAudio 方法，包括 start()方法、pause()方法和 stop()方法。start()方法执行文件；pause()方法暂停播放和 stop()方法停止播放文件。

```
<HTML>
<HEAD><TITLE>LiveAudio Demo</TITLE></HEAD>
<BODY>
<EMBED SRC="sound_file.wav" NAME="audio"
AUTOSTART="true" HIDDEN="true">
<H2>LiveAudio Demo</H2>
<FORM>
<INPUT TYPE="button" VALUE=" Play Audio File "
      onClick="document.audio.start();">
<INPUT TYPE="button" VALUE=" Pause Audio File "
      onClick="document.audio.pause();">
<INPUT TYPE="button" VALUE=" Stop Audio File "
      onClick="document.audio.stop();">
</FORM>
</BODY>
```

图 12-19　用 JavaScript 控制 LiveAudio 插件

RealPlayer 插件包括若干控制文件执行的方法。将用到下面的三种方法：DoPlay()、DoPause()和 DoStop()来控制 RealPlayerPlugin.html 文件的执行。DoPlay()方法播放视频；DoPause()方法暂停视频的播放；DoStop()方法停止视频的播放。

提示：RealPlayer 的开发信息可以在地址 http://www.real.com/devzone 找到。

下面，往 RealPlayerPlugin.html 文件添加插件的控制，控制视频的播放：

1．在文本或 HTML 编辑器中打开 RealPlayerPlugin.html 文件。

2．在<EMBED>标签后面添加下面的表单。表单包括三个按钮：一个按钮用 DoPlay()播放视频；一个按钮用 DoPause()暂停视频的播放；另一个按钮用 DoStop()停止视频的播放。

```
<FORM>
<INPUT TYPE="button" VALUE=" Play Video "
```

```
        onClick="document.hellcat.DoPlay();">
<INPUT TYPE="button" VALUE=" Pause Video "
        onClick="document.hellcat.DoPause();">
<INPUT TYPE="button" VALUE=" Stop Video "
        onClick="document.hellcat.DoStop();">
</FORM>
```

3．保存文件并在 Web 浏览器中打开。测试一下开始、暂停和停止按钮，看它们是否工作正确。图 12-20 显示了该文件在 Navigator 中的输出。

4．关 Web 浏览器窗口。

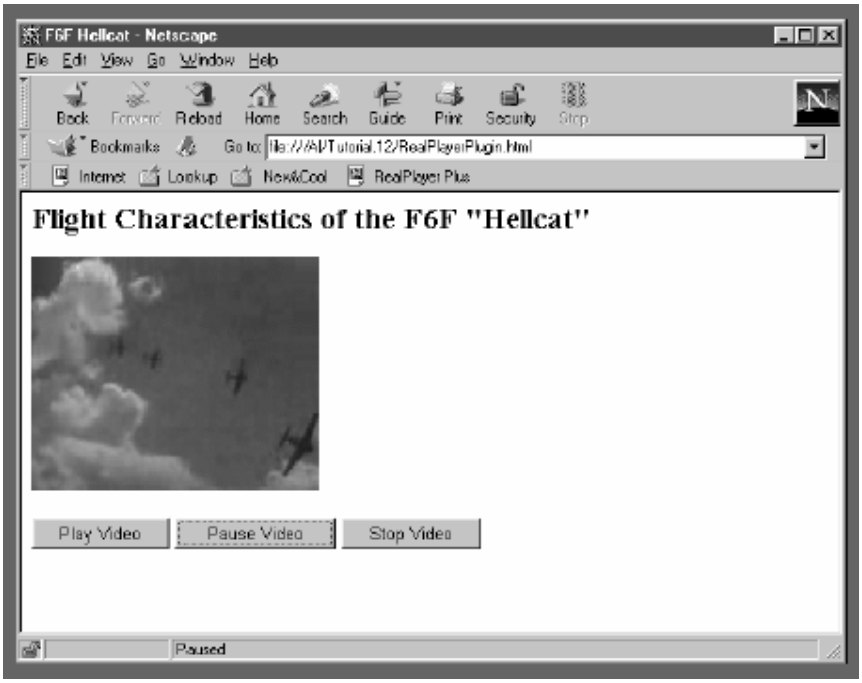


图 12-20　RealPlayerPlugin.html 文件在 Navigator 中的输出

ActiveX 控件

ActiveX 是一种允许编程对象更容易的被重用的技术，ActiveX 允许编程对象更容易地被重用在支持微软的组件对象模型的编程语言中。组件对象模型（Component Object Model，COM）是一种跨平台的客户/服务器应用的体系结构。ActiveX 控件实际上是一些对象，这些对象放置在 Web 页面或支持 COM 的编程语言创建的程序中。用一种脚本语言，例如 JavaScript 或 VBScript 来控制 Web 页面中的 ActiveX 控件，这种脚本语言称为 ActiveX 脚本语言。可以将 ActiveX 控件与插件和 Java 的结合体相比较；ActiveX 可以被用来显示嵌入信息和执行类似于 applets 的小应用程序。ActiveX 在 Windows 编程里是十分常见的。可以在 Internet 上查找到数以千计的 ActiveX 控件，ActiveX 控件应用的普遍性正是微软在 IE

中对其进行支持的主要原因。

提示：许多 ActiveX 控件是需要许可证的，必须从其开发者处进行购买。但是，在 Internet 上，可以找到许多免费的 ActiveX 控件。如果需要查找专业的 ActiveX 控件的信息，请浏览微软的 COM 资源站点 http://www.microsoft.com/com/resources /websites.asp。该站点中还列出了技术信息和培训资源，可以在它们的帮助下开发自己的 ActiveX 控件。

既然 IE 支持插件，那么为什么 ActiveX 控件是必需的呢？例如，RealPlayerPlugin. html 在 IE 和在 Navigator 中运行的同样好。原因在于，与 RealPlayer 不同，许多 ActiveX 控件并不能以插件形式存在。如果用的 ActiveX 控件没有相应的插件，就必须在 Web 页面中直接使用 ActiveX 控件了。在页面中使用 ActiveX 控件，可以极大地增强页面的功能。Navigator 通过 Ncompass 的 ScriptActive 插件同样可以支持 ActiveX 控件。如果要在 Web 页面中使用 ActiveX 控件，要注意的是 ActiveX 控件目前只能运行在 Windows 操作系统中。

提示：ActiveX 控件原来的名称是 OLE 控件。

<OBJECT>标签

标签对<OBJECT>…</OBJECT>用来在 HTML 文档中放置 ActiveX 或其他类型的对象。<OBJECT>标签的四个属性是 ID、CLASSID、HEIGHT 和 WIDTH。ID 属性给对象赋一个惟一的标识，可以用该标识在 JavaScript 中操作该对象，就像<EMBED>标签中的 NAME 属性。CLASSID 属性是用<OBJECT>标签嵌入的对象的一个惟一标识号。所有的 ActiveX 控件都有一个惟一的标识号，该标识号可以在 ActiveX 控件的文档中找到。用 CLASSID 属性指定一个 ActiveX 控件的语句如下所示：CLASSID="clsid:unique id "。举例来说，RealPlayer 的惟一标识是 CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA。要设计一个运行 RealPlayer 的<OBJECT>标签，需要包括下面的代码 CLASSID="clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA"。HEIGHT 和 WIDTH 属性在 Web 页面上为 ActiveX 控件创建一个矩形区域。

要标识 ActiveX 控件的其他属性，需要在<OBJECT>…</OBJECT>标签对中放置<PARAM>标签。每个<PARAM>标签指定控件的一个属性，标签中包括了作为参数名称的 NAME 属性，还包括参数的 VALUE 属性。<PARAM>标签相当于插件的客户属性。可以从 ActiveX 控件的开发者处获得<PARAM>的相关信息。

微软分发了一个称为 ActiveX 控件便笺的免费程序，该程序协助往 Web 页面中添加 ActiveX 控件。如果有一个已经安装在系统中的 ActiveX 控件，需要在 HTML 文档中使用，ActiveX 控件便笺能够将控件的 CLASSID 参数插入到<OBJECT>标签中，并且可以帮助设置控件必需的<PARAM>标签。可以从 http://msdn.microsoft.com/workshop/misc/cpad/ 下载控件便笺程序。

下面，创建一个 ActiveX 版本的 RealPlayerPlugin.html 文件：

1．在文本或 HTML 编辑器中打开 RealPlayerPlugin.html 文件，另存为 RealPlayerActiveX. html 文件，保存在 Tutorial.12 文件夹中。

2．用下面的<OBJECT>…</OBJECT>标签对替代<EMBED>标签。ID 属性赋给一个为 hellcat 的值，将 RealPlayer ActiveX 控件的惟一标识号赋给 CLASSID 属性。在<OBJECT>…</OBJECT>标签对之间还包括两个参数：一个参数指定 SRC 值为 hellcat.rpm，另一个参数指定 CONTROLS 值为 ImageWindow。

```
<OBJECT ID="hellcat"
CLASSID="clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA"
WIDTH=220 HEIGHT=180>
<PARAM NAME="SRC" VALUE="hellcat.rpm">
<PARAM NAME="CONTROLS" VALUE="ImageWindow">
</OBJECT>
```

3．保存文件并在 IE 中打开，进行测试。

4．关闭 Web 浏览器窗口。

12.2.7　总结

- ✧ LiveConnect 包提供了 Navigator 存取核心 Java 的功能，并且包含了 Java 程序控制 JavaScript 的必需的类。
- ✧ CLASSPATH 环境变量告诉 Java 虚拟机和 JDK 应该到哪里去找 Java 应用程序需要的类。在 Windows 95/98 系统中需要在 autoexec.bat 文件中设置 CLASSPATH 环境变量。
- ✧ 因为 Java 是一种类型很强的语言，必须保证 JavaScript 的数据类型正确地转到 Java 变量。
- ✧ 从 JavaScript 传递到 Java 的复杂对象，被封装在 JSObject 封装器中，从 Java 传递 JavaScript 的复杂对象被封装在 JavaObject 封装器中。
- ✧ 封装器是一个类或者对象，封装了其他类或对象以及相关信息。
- ✧ JSObject 类包含了允许 Java 和 JavaScript 之间交互的方法，并且可以作为一个 JavaScript 对象封装器进行工作。
- ✧ JSException 类将 JavaScript 的错误传递回 Java 类。
- ✧ JSObject 和 JSException 两个类都包含在 LiveConnect netscape.javascript 包中。
- ✧ 要用 JSObject 方法存取 JavaScript 程序，首先必须用 getWindows()方法来存取浏览器窗口，操作是通过窗口句柄进行的。一个句柄确定一个操作系统资源。在这种情况下，资源指的是浏览器窗口。
- ✧ 在创建了一个句柄以后，需要在句柄后添加 getMember()方法，以返回对 JavaScript 程序中其他对象的引用。
- ✧ MAYSCRIPT 标签给了一个 applet 在 Web 页面中操作 JavaScript 代码的权限。
- ✧ Packages 对象为 JavaScript 提供了存取包含在 LiveConnect 中的 Java 包、类和方法

的功能。

- ✧ LiveConnect 为 JavaScript 提供了在 Navigator 中操作插件的能力。
- ✧ IE 提供了对插件的支持，但是用 ActiveX 技术替代 LiveConnect 作为底层支持技术。
- ✧ 助手应用程序是一个用来打开 Navigator 不支持的文件类型的外部程序。
- ✧ 插件技术是由第三方开发者开发的软件组件，组件允许在 Navigator 窗口中显示和执行不同类型的嵌入信息。
- ✧ 当存取在服务器上的一个 Web 页面时，Navigator 启动一个插件或助手应用程序，根据页面的 MIME 类型，用指定的媒体类型来打开它。
- ✧ 当安装了一个插件或助手应用程序后，就可以通过添加<EMBED>标签，在 Web 页面中添加嵌入数据。
- ✧ 可以在 JavaScript 中通过嵌入数据的数组 embeds[]或<EMBED>的 NAME 属性来引用嵌入数据。
- ✧ ActiveX 是一种允许编程对象更容易地被重用在支持微软的组件对象模型的编程语言中的技术。
- ✧ ActiveX 控件实际上是一些对象，这些对象放置在 Web 页面或支持 COM 的编程语言创建的程序中。
- ✧ 用一种脚本语言，例如 JavaScript 或 VBScript，来控制 Web 页面中的 ActiveX 控件，这种脚本语言称为 ActiveX 脚本语言。
- ✧ 标签对<OBJECT>…</OBJECT>用来在 HTML 文档中放置 ActiveX 或其他类型的对象。
- ✧ ID 属性给对象赋一个惟一的标识，可以用该标识在 JavaScript 中操作该对象，就像<EMBED>标记中的 NAME 属性。
- ✧ CLASSID 属性是用<OBJECT>标签嵌入的对象的一个惟一标识号。
- ✧ 要标识 ActiveX 控件的其他属性，需要在<OBJECT>…</OBJECT>标签对中放置<PARAM>标签。

12.2.8 问题

1．Java 包和它们的目录结构包含在一个压缩文件中，文件名为：

a. java.zip

b. java.jar

c. classes.jar

d. classes.zip

2．Navigator 4 的 LiveConnect 包包含在压缩文件_____中。

a. java40.jar

b. nav40.jar

c. java40.zip

d. nav40.zip

3．下面哪个环境变量告诉 Java 虚拟机和 JDK 应该到/bin 目录去找 Java 应用程序需要的类？

a. JDKPATH

b. JAVACLASS

c. CLASSPATH

d. JAVAPATH

4．下面哪个类包含了允许 Java 和 JavaScript 之间交互的方法，并且可以作为一个 JavaScript 对象封装器进行工作？

a. JSObject

b. JavaToJavaScript

c. JavaScriptObjects

d. JavaScriptJava

5．下面哪个语句将 LiveConnect 包引入 Java 程序？

a. netscape.javascript.*;

b. netscape.*;

c. netscape.packages.*;

d. netscape.liveconnect.*;

6．下面哪个是获得一个窗口句柄的正确语法？

a. windowVar = JSObject.getWindow(this);

b. JSObject windowVar = getWindow(this);

c. windowVar JSObject = JSObject.getWindow(this);

d. JSObject windowVar = JSObject.getWindow(this);

7．下面哪个语句返回对一个窗口的文档对象的引用？假设已经声明了一个 windowVar 句柄。

a. JSObject windowVar = (JSObject)
jsWindow.getMember("document");

b. windowVar = (JSObject) jsWindow.getMember("document");

c. JSObject windowVar = jsWindow.getMember("document");

d. JSObject windowVar = getMember(jsWindow.Document);

8．应该如何将一个 JavaScript 表单元素的值赋给一个 Java 字符串变量？假设已经创建了一个 textField 文本域。

a. String javaStringVar = textField.getMember("value");

b. String javaStringVar = (String) textField.getMember.value;

c. String javaStringVar = textField (getMember.value);

d. String javaStringVar = (String) textField.getMember("value");

9．在 Java 程序中执行 JavaScript 的警告对话框，下面哪个是正确的语法？假设已经声明了一个 windowVar 句柄。

a. windowVar.eval(alert("Text String"););

b. windowVar.eval.alert("Text String");

c. windowVar.eval(“alert('Text String');");

d. windowVar.alert("Text String").eval;

10．下面哪个属性使 Java 程序可以存取 JavaScript 和 HTML 文档中的元素？

a. SCRIPT="true"

b. MAYSCRIPT

c. JAVA="true"

d. SCRIPTALLOWED

11．下面的哪个语句可以让 Navigator 中的 JavaScript 程序执行 Java.lang.Math 类中的 round()方法？

a. Packages.java.lang.Math.round();

b. Java.java.lang.Math.round();

c. Packages.Math.round();

d. Packages.java.Math.round();

12．Ncompass 中的哪个插件允许 Navigator 使用 ActiveX 控件？

a. ScriptActive

b. ActiveScript

c. JavaScriptActive

d. NavigatorActiveX

13．插件技术是由_____开发的允许在 Navigator 窗口中显示和执行不同类型的嵌入信息的组件。

a. Microsoft

b. Netscape

c. Sun Microsystems

d. 第三方开发者

14．嵌入数据通过它的 MIME 类型或_____与一个指定插件关联。

a. 在 SRC 属性中指定的 HTTP 协议

b. 在 NAME 属性中赋给的取值

c. 目录定位

d. 文件扩展名

15．用_____标签将嵌入数据添加到一个 Web 页面中。

a. <LINK>

b. <OBJECT>

c. <PLUG-IN>

d. <EMBED>

16. 在 JavaScript 中，可以用 NAME 属性或_____来引用嵌入数据。

a. 直接存取它的方法和属性

b. 将它的 SRC 属性添加到 java.Objects 类中

c. 在 embeds[]数组中引用它对应的元素

d. 在 JavaScript 中没法引用嵌入数据

17. 下面哪个平台允许使用 ActiveX 控件？

a. Windows

b. Macintosh

c. Windows and Macintosh

d. UNIX

18. 用_____标签可以将 ActiveX 控件添加到一个 Web 页面中。

a. <LINK>

b. <OBJECT>

c. <PLUG-IN>

d. <EMBED>

19. 在 JavaScript 中通过_____可以引用一个嵌入的 ActiveX 控件。

a. ID 属性

b. NAME 属性

c. objects[]数组中的对应元素

d. 类名

20. 系统定位一个 ActiveX 控件需要的惟一的标识号称为_____。

a. DATAID

b. CONTROLID

c. ACTIVEXID

d. CLASSID

21. 如何将取值传递给一个 ActiveX 控件？

a. 通过各种属性

b. 通过<PARAM>标签

c. 为 SRC 属性添加一个文本字符串

d. 没法往 ActiveX 控件添加取值

22. 微软分发的，协助往 Web 页面中添加 ActiveX 控件的免费程序称为：

a. 嵌入工具

b. ActiveX 助手

c. ActiveX 控件便笺

d．ActiveX 工具箱

12.2.9 练习

1. 创建一个带表单的 HTML 文档，表单中包括了最近的三个工作的信息。包括雇主姓名的列表选择域、工资和在那里工作的时间。然后，创建一个 Java 小应用程序读取表单中的工资和工作时间域，并且使用 Java 的 Math.max()方法计算最高的工资和最长的工作时间。并且用 Java 的 Math.min()方法计算最低的工资和最短的工作时间。计算完成后，在 applet 的区域中显示这些信息。保存文件为 WorkHistory.html 和 WorkHistory.java。如果换了不到三个工作，假想几个名字和工资。

2. 用 Netscape 提供的直接 Java 调用功能创建一个更高级的计算器程序。在第 3 章曾经创建过该程序。参考 JDK 的文档，在 Math 类中查找可能在计算器程序中用到的方法。例如 exp()和 sqrt()方法。注意只有 Navigator 支持直接 Java 调用。如果没有 Navigator，则将计算器的计算部分创建成一个 Java 小应用程序，用 HTML 表单输入数值和选择操作。

3. 在 Internet 上查找关于如何创建插件的信息，并以此为题写一篇短论文。

4. 微软的站点包含了许多关于创建 ActiveX 控件的文章和信息。利用这些信息，写一篇论述 COM 和 ActiveX 历史和发展的论文。

5. 写一篇论文，讨论各种可以用来创建 ActiveX 控件的编程语言的不同之处。特别要指出如何让 ActiveX 和 Java 编程集成在一起。

附录 A　JavaScript 参考

注释类型

行注释

```
<SCRIPT LANGUAGE="JavaScript1.2" >
// Line comments are preceded by two slashes.
</SCRIPT>
```

块注释

```
<SCRIPT LANGUAGE="JavaScript1.2" >
/*
This line is part of the block comment.
This line is also part of the block comment.
*/
/* This is another way of creating a block comment. */
</SCRIPT>
```

JavaScript 保留字

abstract	else	int	switch
boolean	extends	interface	synchronized
break	false	long	this
byte	final	native	throw
case	finally	new	throws
catch	float	null	transient
char	for	package	true
class	function	private	try
const	goto	protected	typeof
continue	if	public	var
default	implements	return	void
delete	import	short	while
do	in	static	with
double	instanceof	super	

标识符

合法标识符

my_identifier
$my_identifier
_my_identifier
my_identifier_example
myIdentifierExample

非法标识符

%my_identifier
1my_identifier
#my_identifier
@my_identifier
~my_identifier
+my_identifier

事件

JavaScript 事件

事　　件	触 发 时 间
abort	中断载入图像时
blur	元素（如单选按钮）不可用时
click	单击元素一次时
change	元素值改变时
error	在载入文档或图像出现错误时
focus	激活元素时
load	文档或图像载入时
mouseOut	鼠标移出元素时
mouseOver	鼠标在元素上移动时
reset	重置表单时
select	用户选中表单上的域时
submit	用户提交表单时
unload	文档卸载时

HTML 元素和相关的 JavaScript 事件

元　素	描　述	事　件
<A>...</A>	链接	click mouseOver mouseOut
<IMG>	图像	abort error load
<AREA>	区域	mouseOver mouseOut
<BODY>...</BODY>	文档体	blur error focus load unload
<FRAMESET>...</FRAMESET>	帧集	blur error focus load unload
<FRAME>...</FRAME>	帧	blur focus
<FORM>...</FORM>	表单	submit reset
<INPUT TYPE="text">	文本框	blur focus change select
<TEXTAREA>...</TEXTAREA>	文本域	blur focus change select
<INPUT TYPE="submit">	提交	click
<INPUT TYPE="reset">	重置	click
<INPUT TYPE="radio">	单选按钮	click
<INPUT TYPE="checkbox">	检查框	click
<SELECT>...</SELECT>	组合框	blur focus change

基本数据类型

数 据 类 型	描　　述
整型	不带有小数部分的正数或负数
浮点型	带有小数部分的正数或负数，或用指数符号书写的数
布尔	true 或 false 逻辑值
字符串	文本，如“ Hello World ”
Null	空值

JavaScript 转义序列

转 义 序 列	字　　符
\b	退格
\f	换页
\n	换行
\r	回车
\t	水平制表符
\'	单引号
\"	双引号
\\	反斜杠

数据类型转换函数和方法

语　　法	函数或方法	描　　述
parseFloat(变量);	parseFloat()函数	将字符串转换为浮点数
parseInt(变量);	parseInt()函数	将字符串转换为整数
变量.toString();	toString()方法	将对象值或数转换为字符串
对象.valueOf();	valueOf()方法	返回对象的原值

运算符

JavaScript 运算符

运算符类型	描　　述
算术	用来执行算术运算
赋值	将值赋给变量

续表

运算符类型	描　　述
比较	比较操作数并返回布尔值
逻辑	用来对布尔操作数进行布尔运算
字符串	对字符串进行运算

算术二元运算符

运　算　符	描　　述
+（加）	将两个操作数相加
–（减）	从一个操作数中减去另一个操作数
*（乘）	一个操作数与另一个操作数相乘
/（除）	一个操作数除以另一个操作数
%（取模）	将两数相除并返回余数

算术单目运算符

运　算　符	描　　述
++（自增）	将操作数加 1
––（自减）	将操作数减 1
–（取负）	返回操作数的相反数（负或正）

赋值运算符

运　算　符	描　　述
=	将右操作数赋给左操作数
+=	将右操作数的值与左操作数的值相加，或将右操作数的值加到左操作数上并将新值赋给左操作数
–=	将左操作数的值减去右操作数的值并将新值赋给左操作数
*=	将左操作数的值乘以右操作数的值并将新值赋给左操作数
/=	将左操作数的值除以右操作数的值并将新值赋给左操作数
%=	取模——将左操作数的值除以右操作数的值并将余数赋给左操作数

关系运算符

运　算　符	描　　述
==（等于）	若操作数相等返回真
!=（不等于）	若操作数不相等返回假
>（大于）	若左操作数大于右操作数返回真

续表

运 算 符	描 述
<（小于）	若左操作数小于右操作数返回真
>=（大于或等于）	若左操作数大于或等于右操作数返回真
<=（小于或等于）	若左操作数小于或等于右操作数返回真

逻辑运算符

运 算 符	描 述
&&（与）	若左操作数和右操作数都为真，那么就返回真；否则返回假
||（或）	若左操作数或右操作数中一个为真，那么就返回真。若都不为真，那么含有||（或）运算符的表达式将返回假
!（非）	若表达式为假则返回真；若表达式为真则返回假

运算符优先级

- ✧ 括弧（（）、[ ]、.） 优先级最高
- ✧ 取反/自增（!、−−、++、−、typeof、void）
- ✧ 乘/除/取模（*、/、%）
- ✧ 加/减（+、−）
- ✧ 比较（<、<=、>、>=）
- ✧ 相等（==、!=）
- ✧ 逻辑与（&&）
- ✧ 逻辑或（||）
- ✧ 赋值运算符（=、+=、−=、*=、/=、%=） 优先级最低

控制结构和语句

```
if  (条件表达式 ) {
        语句(s);
}
if  (条件表达式) {
        语句(s);
}
else  {
        语句(s);
}

switch  (表达式) {
        case 标号 :
              语句(s);
            break;
```

```
        case 标号 :
             语句(s);
           break;
       ...
        default :
             语句(s);
}
while  (条件表达式) {
        语句(s);
}
do  {
        语句(s);
}  while  (条件表达式);
for  (初始化表达式;  条件 ;  更新语句)  {
        语句(s);
}
for  (变量 in 对象 ) {
        语句(s);
}
with  (对象) {
        语句(s);
}
```

break　break 语句用来跳出 switch 语句和其他程序控制语句如 while、do…while、for 和 for…in 循环语句。为了在 switch 语句完成所需的任务之后就终止它，应该在每个 case 标号内包括 break 语句。

continue　continue 语句中断循环语句并以新循环值重新开始循环。在想中断当前循环并想以新值继续循环时请使用 continue 语句。

对象

本节列出了主要的 JavaScript 对象的属性、方法和事件。它仅列出了与 Internet Explorer 和 Navigator 都兼容的属性、方法和事件。

Date 对象

方　法	描　述
getDate()	返回 Date 对象的日期
getDay()	返回 Date 对象的某一天
getFullYear()	以 4 位数格式返回 Date 对象的年份
getHours()	返回 Date 对象的小时
getMilliseconds()	返回 Date 对象的毫秒数
getMinutes()	返回 Date 对象的分钟

续表

方　法	描　述
getMonth()	返回 Date 对象的月份
getSeconds()	返回 Date 对象的秒数
getTime()	返回 Date 对象的时间
getTimezoneOffset()	以分钟返回当前日期和 GMT 之间的本地时差
getUTCDate()	以通用时间格式返回 Date 对象的日期
getUTCFullYear()	以通用时间格式返回 Date 对象的 4 位数年份
getUTCHours()	以通用时间格式返回 Date 对象的小时数
getUTCMilliseconds()	以通用时间格式返回 Date 对象的毫秒数
getUTCMinutes()	以通用时间格式返回 Date 对象的分钟数
getUTCMonth()	以通用时间格式返回 Date 对象的月份
getUTCSeconds()	以通用时间格式返回 Date 对象的秒数
setDate()	设置 Date 对象的日期
setFullYear()	设置 Date 对象的 4 位数年份
setHours()	设置 Date 对象的小时数
setMilliseconds()	设置 Date 对象的毫秒数
setMinutes()	设置 Date 对象的分钟数
setMonth()	设置 Date 对象的月份
setSeconds()	设置 Date 对象的秒数
setTime()	设置 Date 对象的时间
setUTCDate()	以通用时间格式设置 Date 对象的日期
setUTCFullYear()	以通用时间格式设置 Date 对象的 4 位数年份
setUTCHours()	以通用时间格式设置 Date 对象的小时数
setUTCMilliseconds()	以通用时间格式设置 Date 对象的毫秒数
setUTCMinutes()	以通用时间格式设置 Date 对象的分钟数
setUTCMonth()	以通用时间格式设置 Date 对象的月份
setUTCSeconds()	以通用时间格式设置 Date 对象的秒数
toGMTString()	将 Date 对象转换为 GMT 时区的格式的字符串
toLocaleString()	将 Date 对象转换为当前时区的格式的字符串
toString()	将 Date 对象转换为字符串
toUTCString()	将 Date 对象转换为通用时间的格式的字符串
valueOf()	将 Date 对象转换为毫秒格式

Document 对象

属　性	描　述
alinkColor	激活链接的颜色，由<BODY>标签的 ALINK 属性指定

续表

属　性	描　述
anchors[]	引用文档锚的数组
applets[]	引用文档的 applet 的数组
bgColor	文档背景色，由<BODY>标签的 BGCOLOR 属性指定
cookie	为当前文档指定 cookie
domain	当前文档所在的服务器的域名
embeds[]	引用文档的插件和 ActiveX 控件的数组
fgColor	文档文本的前景色，由<BODY>标签的 FGCOLOR 属性指定
forms[]	引用文档的表单的数组
images[]	引用文档的图像的数组
lastModified	文档上次被修改的日期
linkColor	文档未激活链接的颜色，由<BODY>标签的 LINK 属性指定
links[]	引用文档链接的数组
referrer	提供了链接当前文档的 URL
title	文档标题，由文档<HEAD>段内的<TITLE>…</TITLE>标签对指定
URL	当前文档的 URL
vlinkColor	文档已浏览的链接的颜色，由<BODY>标签的 VLINK 属性指定

方　法	描　述
close()	通知 Web 浏览器已写完窗口或帧的内容，应该显示文档了
open()	打开另一个窗口或帧，而不是当前窗口或帧，并与 write()和 writeln()方法一起被用来更新其内容
write()	在 Web 页面上创建新文本
writeln()	在 Web 页面上创建新文本，紧跟一个换行

Form 对象

属　性	描　述
action	将表单数据提交给它的 URL
method	以何种方法提交表单数据：GET 或 POST
enctype	数据被提交的格式
target	在其中显示服务器所返回的任何结果的窗口
name	表单名
elements[]	代表表单元素的数组
length	表单上元素数

方　法	描　述
reset()	清除在表单内所输入的数据

续表

方　法	描　述
submit()	将表单提交给 Web 服务器

事　件	描　述
onReset	reset 按钮被按下或 reset()方法被调用
onSubmit	submit 按钮被按下或 submit()方法被调用

Frame 对象

属　性	描　述
action	将表单数据提交给它的 URL
method	以何种方法提交表单数据：GET 或 POST
enctype	数据被提交的格式
target	在其中显示服务器所返回的任何结果的窗口
name	表单名
elements[]	代表表单元素的数组
length	表单上元素数

方　法	描　述
reset()	不使用 reset <INPUT>标签清除表单
submit()	不使用 submit <INPUT>标签将表单提交给 Web 服务器

事　件	描　述
onReset	在表单上的 reset 按钮被选中时执行
onSubmit	在使用 submit <INPUT>标签或 image <INPUT>标签提交表单时执行

History 对象

属　性	描　述
length	当前浏览器会话过程中已被打开的、指定的文档数

方　法	描　述
back()	与单击 Web 浏览器的“后退”按钮等同
forward()	与单击 Web 浏览器的“前进”按钮等同
go()	打开历史列表内指定的文档

Image 对象

属　性	描　述
border	只读属性，用像素表示的边界宽度，值与<IMG>标签的 BORDER 属性所指定的相同
complete	布尔值，在载入整幅图像时返回 true
height	只读属性，用像素表示的图像高度，值与<IMG>标签的 HEIGHT 属性所指定的相同
hspace	只读属性，用像素表示的水平空白数，值与<IMG>标签的 HSPACE 属性所指定的相同
lowsrc	用来显示低分辨率的可选图像的 URL
name	赋给标签的名称
src	所显示图像的 URL
vspace	只读属性，用像素表示的垂直空白数（图像的顶部和底部），值与<IMG>标签的 VSPACE 属性所指定的相同
width	只读属性，图像宽度，值与<IMG>标签的 WIDTH 属性所指定的相同

事　件	描　述
onLoad	完成载入图像
onAbort	用户取消载入图像，通常是单击“停止”按钮
onError	在载入图像时出现错误

Location 对象

属　性	描　述
hash	URL 的锚
host	URL 的主机名和端口部分
hostname	URL 主机名
href	完整的 URL 地址
pathname	URL 路径
port	URL 端口
protocol	URL 协议
search	URL 搜索或查询部分

方　法	描　述
reload()	再次打开 Web 浏览器内当前显示的页面
replace()	用一个不同 URL 代替当前载入的 URL

Math 对象

属　性	描　述
E	欧拉常数 e，它是自然对数的底数
LN10	自然对数 10
LN2	自然对数 2
LOG2E	底为 2 的对数 e
LOG10E	底为 10 的对数 e
PI	代表圆周长和半径比的常数
SQRT1_2	2 的平方根的倒数
SQRT2	2 的平方根

方　法	描　述
abs(x)	返回 x 的绝对值
acos(x)	返回 x 的反余弦值
asin(x)	返回 x 的反正弦值
atan(x)	返回 x 的反正切值
atan2(x,y)	返回 x 轴的角度
ceil(x)	将 x 四舍五入，值为大于或等于 x 的最小整数
cos(x)	返回 x 的余弦
exp(x)	返回 x 的指数
floor(x)	将 x 四舍五入，值为大于或等于 x 的最大整数
log(x)	返回 x 的自然对数
max(x,y)	返回两数之间的大者
min(x,y)	返回两数之间的小者
pow(x,y)	返回 x 的 y 次幂
random()	返回一个随机数
round(x)	返回 x 四舍五入后的值
sin(x)	返回 x 的正弦值
sqrt(x)	返回 x 的平方根
tan(x)	返回 x 的正切值

Navigator 对象

属　性	描　述
appCodeName	Web 浏览器代码名
appName	Web 浏览器名
appVersion	Web 浏览器版本

续表

属　性	描　述
language	Web 浏览器所使用的语言，如 English 或 French
platform	所用的操作系统
userAgent	用户代理

方　法	描　述
javaEnabled()	确定在当前浏览器中是否启动 Java

String 对象

方　法	描　述
anchor(锚名)	将<ANCHOR>…</ANCHOR>标签对添加到文本字符串内
big()	将<BIG>…</BIG>标签对添加到文本字符串内
blink()	将<BLINK>…</BLINK>标签对添加到文本字符串内
bold()	将<B>…</B>标签对添加到文本字符串内
charAt(索引)	返回文本字符串内指定位置处的字符。若指定位置大于字符串的长度，则返回空
fixed()	将<TT>…</TT>标签对添加到文本字符串内
fontcolor(颜色)	将<FONT COLOR=颜色>…</FONT>标签对添加到文本字符串内
fontsize(大小)	将<FONT SIZE=大小>…</FONT>标签对添加到文本字符串内
indexOf(文本, 索引)	返回文本参数的第一个字符在字符串内的位置。若包括了索引参数，那么 indefOf()方法将从那个位置起开始在字符串内搜索。若未找到字符或字符串，那么返回−1
italics()	将<I>…</I>标签对添加到文本字符串内
lastIndexOf(文本, 索引)	返回文本参数内的第一个字符在字符串内最后出现一次的位置。若含有索引参数，那么 lastIndexOf()方法从字符串的此位置器开始搜索。若字符或字符串未找到则返回-1
link(*href*)	将<A HREF=*URL*>…</A>标签对添加到文本字符串内
small()	将<SMALL>…</SMALL>标签对添加到文本字符串内
split(*separator*)	按指定条件将文本字串分成子串
strike()	将<STRIKE>…</STRIKE>标签对添加到文本字符串内
sub()	将_…标签对添加到文本字符串内
substring(起始索引，结束索引)	从索引文本字符串的指定位置开始到结束位置至取子字符串

续表

方　法	描　述
sup()	将[…]标签对添加到文本字符串内
toLowerCase()	将指定的文本字符串转换为小写
toUpperCase()	将指定的文本字符串转换为大写

属　性	描　述
length	返回字符串内的字符数

目标及相关权限

属　性	描　述
UniversalBrowserRead	检索 History 对象的 any 属性值 检索 DragDrop 事件的 data 属性值 使用 about:URL 而不是使用 about:blank
UniversalBrowserWrite	使用 Window 对象的 enabelExternalCapture()方法来捕获从不同服务器上载入的页面内的事件 使用 Window 对象的 moveBy()或 moveTo()方法将窗口移出屏幕外 读取或设置 Event 对象的 any 属性 读取或设置 Window 对象的所有下列属性：locationbar、menubar、personalbar、scrollbars、statusbar 和 toolbar 使用 Window 对象的 close()方法无条件地关闭浏览器窗口 使用 Window 对象的 innerWidth()或 innerHeight()方法将窗口的内部宽度和高度设置为比 100×100 小或比能够容纳它的屏幕大的尺寸 使用 Window 对象的 open()方法来创建一个不具有标题栏的窗口 使用 Window 对象的 open()方法将窗口放置在屏幕外 在 Window 对象的 open()方法内使用 alwaysLowered、alwaysRaised 或 z-lock 使用 Window 对象的 open()方法创建一个比 100×100 小或比能够容纳它的屏幕大的窗口 使用 Window 对象的 resizeTo()或 resizeBy()方法将窗口的尺寸调整为比 100×100 小或比能够容纳它的屏幕大的尺寸
UniversalFileRead	读取 FileUpload 对象的 value 属性
UniversalPreferencesRead	使用 Navigator 对象的 preferences()方法读取所有引用
UniversalPreferencesWrite	使用 Navigator 对象的 preferences()方法写入所有引用
UniversalSendMail	将表单提交给 mailto:或 news:URL

Windows 对象

属　性	描　述
defaultStatus	写在状态条上的默认文本
document	Document 对象的引用
frames[]	一个列出了窗口内帧对象的数组
history	对 History 对象的引用
location	对 Location 对象的引用
name	窗口名
opener	打开另一个窗口的 Window 对象
parent	包含当前帧的父帧
self	Window 对象的自我引用——与 window 属性相同
status	写在状态条上的临时文本
top	包含了当前帧的最顶层 Window 对象
window	Window 对象的自我引用——与 self 属性相同

方　法	描　述
alert()	显示一个简单的、具有“OK”按钮的消息对话框
blur()	从窗口移去焦点
clearTimeout()	取消设置的超时
close()	关闭窗口
confirm()	显示一个具有“OK”和“Cancel”按钮的确认对话框
focus()	让 Window 对象成为活动窗口
open()	打开一个新窗口
prompt()	显示一个对话框提示用户输入信息
setTimeout()	在过去指定的秒数之后执行一个函数

事　件	描　述
onBlur	窗口变为非活动
onError	在窗口载入时发生错误
onFocus	窗口变为活动
onLoad	在窗口载入全部文档
onResize	窗口改变尺寸
onUnload	卸载窗口内的当前文档

附录 B　LiveWire 参考

LiveWire 全局函数

函　　数	描　　述
write()	向客户返回文本
flush()	清空输出缓冲区
redirect()	将客户重定向至指定的 URL
getOptionValue()	返回 HTML <OPTION>标签的单个选项值
getOptionValueCount()	返回 HTML SELECT 标签内<OPTION>标签数
debug()	在一个帧内显示表达式的输出结果
addClient()	将客户信息添加在 URL 后
registerCFunction()	准备在 LiveWire 中使用 C 函数
callC()	执行 C 函数
deleteResponseHeader()	删除从客户发送来的响应首部内的信息
addResponseHeader()	将信息添加在从客户发送来的响应后
ssjs_getClientID()	返回某些客户维护技术所使用的客户 ID
ssjs_generateClientID()	创建一个惟一地标识客户对象的标识符
ssjs_getCGIVariable()	返回一个指定的 CGI 环境变量的值

预定义对象

Cursor 对象

方　　法	描　　述
close()	关闭 Cursor 对象
columnName(位置)	返回位置参数所指定的列名
columns()	返回记录集的列数
deleteRow()	从记录集中删除一行
insertRow()	往记录集中插入一行
next()	移至记录集中下一条记录
updateRow()	保存对记录集中当前行的修改

Database 和 Connection 对象

方　法	描　述
beginTransaction()	开始 SQL 事务
commitTransaction()	保存当前 SQL 事务
connect()	将 LiveWire 与数据库连接
connected()	若成功连接了数据库，那么将返回真
cursor()	为指定的 SQL 语句创建一个数据库游标
disconnect()	关闭数据库连接
execute()	将 SQL 语句发送给数据库管理系统进行处理
majorErrorCode()	返回主要的 ODBC 或数据库服务器产生的错误码
majorErrorMessage()	返回主要的 ODBC 或数据库服务器产生的错误消息
minorErrorCode()	返回次要的 ODBC 或数据库服务器产生的错误码
minorErrorMessage()	返回次要的 ODBC 或数据库服务器产生的错误消息
rollbackTransaction()	回滚当前 SQL 事务
SQLTable()	执行 SQL 语句并将结果作为 HTML 表返回

Request 对象

属　性	描　述
agent	客户浏览器名和版本号
ip	客户 IP 地址
method	请求的 HTTP 方法
protocol	客户浏览器所支持的 HTTP 协议
imageX	在用户单击图像映射之后，光标的水平位置
imageY	在用户单击图像映射之后，光标的垂直位置

Server 对象

属　性	描　述
hostname	完整的主机名
host	名称、子域和域
protocol	通信协议
port	所使用的端口号
jsVersion	版本和平台

状态码

状 态 码	描 述
0	无错
1	内存不足
2	未初始化对象
3	类型转换错误
4	未注册数据库
5	服务器报出的错误
6	来自服务器的消息
7	来自厂商库的错误
8	丢失连接
9	读取结束
10	无效的对象使用
11	列不存在
12	对象内的无效定位（绑定错误）
13	不支持的特性
14	无效的引用参数
15	未发现数据库对象
16	所需信息丢失
17	对象不支持多个访问
18	对象不支持删除
19	对象不支持插入
20, 21	对象不支持更新
22	对象不支持索引
23	对象不能被停止
24	所提供的连接不正确
25	对象不支持权限
26	对象不支持游标
27	不能打开

附录 C　Active Server Pages 参考

处理命令属性

属　　性	描　　述
CODEPAGE	指定 ASP 文档的字符集
ENABLESESSIONSTATE	确定 ASP 文档是否维护状态信息
LANGUAGE	设置文档的默认脚本语言
LCID	设置脚本的本地化标识符
TRANSACTION	确定脚本是否作为事务对待

预定义对象

Request 对象

集　　合	描　　述
ClientCertificate	随请求一起发送的客户认证内的字段值
Cookies	随请求一起发送的 Cookies
Form	浏览器内所显示文档内命名表单元素的值
QueryString	被添加在查询字符串内 URL 后的“名称=值”对
ServerVariables	环境变量

Session 对象

属　　性	描　　述
CodePage	用于指定语言的字符集
LCID	标识用户的区域的或应用程序的首选语言
SessionID	用户会话标识
Timeout	Session 对象的生存期

Server 对象

方　　法	描　　述
CreateObject()	初始化服务器组件

续表

方　法	描　述
Execute()	执行 ASP 文件
GetLastError()	返回 ASPError 对象，它描述了所发生的错误状态
HTMLEncode()	使用 HTML 编码对指定的字符串进行编码
MapPath()	将一个相对的或虚拟的路径映射为服务器相应的物理目录
Transfer()	将当前 ASP 应用程序的集合变量和对象发送给另一个 ASP 应用程序
URLEncode()	使用 URL 编码格式化一个字符串

Connection 对象

方　法	描　述
Open()	打开数据源连接
Close()	关闭数据源连接
Execute()	执行对数据源操作的命令
BeginTrans()	开始事务
CommitTrans()	提交事务
RollbackTrans()	回滚事务

属　性	描　述
ConnectionString	Open()方法的连接字符串参数
ConnectionTimeout	在放弃一个连接尝试前所等待的秒数
CommandTimeout	在放弃一个命令尝试前所等待的秒数
State	数据源的连接状态：adStateOpen 表示一个成功的连接；adStateClosed 表示一个失败的连接
Provider	连接中所使用的数据源
Version	ADO 的版本号
CursorLocation	所返回的值表示是客户管理游标、服务器管理游标还是不管理游标

Recordset 对象

属　性	描　述
ActiveConnection	用于打开 Recordset 对象的数据库连接
BOF	若游标位于文件的开始处则返回真，否则返回假
EOF	若游标位于文件的结束处则返回真，否则返回假

续表

属　性	描　述
CursorLocation	可以设置为决定在何处管理游标的三种值之一：adUseNone、adUseClient 或 adUseServer。adUseClient 表示客户管理游标；adUseServer 表示服务器管理游标；adUseNone 表示不管理游标
CursorType	记录集中所使用的游标类型：adOpenForwardOnly、adOpenKeyset、adOpenDynamic 或 adOpenStatic
MaxRecords	查询所返回的最大记录数
RecordCount	记录集中记录数
Source	用于检索数据集的 SQL 字符串

方　法	描　述
AddNew()	创建新记录
CancelUpdate()	取消任何未决的对记录的修改
Close()	关闭 Recordset 对象
Delete()	删除当前记录
Move()	移至指定的记录号处
MoveFirst()	移至第一条记录处
MoveLast()	移至最后一条记录处
MoveNext()	移至下一条记录处
MovePrevious()	移至上一条记录处
Open()	为 Recordset 打开一个游标
Requery()	重新执行查询刷新 Recordset 对象内的数据
Resync()	不重新执行查询而刷新 Recordset 对象内的数据
Save()	将 Recordset 对象数据保存在文件中
Seek()	在 Recordset 对象中搜索符合指定规则的记录
Supports()	返回 Recordset 对象所支持的功能类型
Update()	保存任何未决的对记录的修改

ADO 参考

对象模型

对　象	描　述
Connection	提供对数据源的访问
Command	执行一条命令（如 SQL 命令）对数据源进行操作
Parameter	为 Command 对象执行的命令提供参数

续表

对　　象	描　　述
Recordset	根据 SQL 查询的结果创建一个可浏览的虚拟表
Field	代表 Recordset 对象的数据库字段
Error	代表数据库所返回的错误
Property	代表 ADO 对象属性

集合

集　　合	描　　述
Errors	数据库返回的错误
Parameters	与 Parameter 对象相关联的参数
Fields	与 Field 对象相关的数据库字段
Properties	ADO 对象属性

游标类型

游 标 类 型	描　　述
adOpenForwardOnly	前向游标。此游标仅让使用 MoveNext()方法向前移动遍历记录集
adOpenKeyset	键集游标。让向前和向后移动遍历记录集，但是不让看到其他用户所添加的新记录。能够看到其他用户所做的任何修改，但是不能访问其他用户已删除的记录
adOpenDynamic	动态游标。允许向前和向后移动遍历记录集并且能够看到其他用户所做的任何修改
adOpenStatic	静态游标。允许向前和向后移动遍历记录集，但是不允许看到其他用户所做的任何修改

锁定类型

锁 定 类 型	描　　述
adLockReadOnly	只读
adLockPessimistic	悲观锁定，一条记录接一条记录锁定
adLockOptimistic	乐观锁定，一条记录接一条记录锁定
adLockBatchOptimistic	乐观批更新

附录 D　Java 参考

基本数据类型

数 据 类 型	描　　述	实　　例
boolean	true 或 false 逻辑值	true 或 false
byte	8 位整数	-128 和 127 之间的值
char	包含在单引号内的任何一个字符或一个数字 Unicode 字符	“A”、“B”、“C”等字符，A、B 和 C 用 Unicode 表示分别是 65、66 和 67
double	64 位浮点数	在 5e-324 和 1.796931348623157e+308 之间的值
float	32 位浮点数	在 1.4e-45f 和 3.4028235e+38f 之间的值
int	32 位整数	在-2147483648 和 2147483647 之间的值
long	64 位整数	在-9223372036854775808 和 9223372036854775807 之间的值
short	16 位整数	在-32768 和 32767 之间的值

JavaScript 和 Java 之间的数据类型转换

JavaScript 数据类型	Java 数据类型
数组	Array 对象
boolean	boolean
浮点数或整数	byte、char、short、int、long、float 和 double
null	null
未定义	void
所有其他 JavaScript 对象	JSObject 封装器
JavaObject 封装器	所有其他 Java 对象

JSObject 类方法

方　法	描　述
call(String 函数名, Object args[])	执行 JavaScript 方法或函数
eval(String 表达式)	执行 JavaScript 字符串表达式
getMember(String 属性名)	返回 JavaScript 对象属性值
getSlot(int 索引)	返回 JavaScript 对象内的数组元素
getWindow(applet)	获取含有 applet 的窗口的句柄
removeMember(String 属性名)	删除 JavaScript 对象内的一个属性
setMember(String 属性名)	设置 JavaScript 对象内的属性
setSlot(int 索引, Object 值)	设置 JavaScript 对象内的数组元素
toString()	返回 JavaScript 对象内的字符串值

读者意见反馈卡

请您认真填写本卡并寄给我们。对于发现本书中技术问题的读者，我们另有答谢。

1．您对本书的总体感觉：

□满意　　□一般　　□不满意

2．您认为本书的层次结构：

□很好　　□一般　　□不好

3．您认为本书的语言文字水平：

□很好　　□一般　　□不好

4．您认为本书的版式编排：

□很好　　□一般　　□不好

5．您认为本书中所涉及各项操作说明的准确性：

□准确　　□较准确　　□不准确

6．您最需要哪方面的图书？ _____

7．您是从哪里第一次听说这本书的？

□书店　　□广告　　□从朋友、同事等处听说　　□其他

8．您一年中购买计算机类图书的数量：

□2～5 本　　□6～10 本　　□多于 10 本

9．您使用的操作系统是：

□DOS　　□Windows　　□OS/2　　□Macintosh　　□Unix　　□Linux　　□其他

10．您感兴趣的计算机类新书为：

□操作系统类　　□办公软件类　　□程序设计语言类

□图形、图像设计类　　□排版软件类　　□网络技术类

□多媒体制作类　　□其他

11．您使用 PC 机的地方：

□家庭　　□单位　　□学校　　□其他

12．您是否有 CD-ROM：

□有　　□无

读者姓名：　　单位名称：　　联系电话：

请填好本卡后寄给：

清华大学校内金地公司(E-mail：jdsd@public.east.net.cn)

《SolidWorks 零件设计》编辑部收

邮编：100084

联系电话：（010）62770381　　传真：（010）62788903

公司网址：www.thjd.com.cn

如需本书可与本编辑部联系邮购。汇款地址请按以上地址填写，另加邮费 15%（挂号）